Aalto University
School of Electrical Engineering
Department of Communications and Networking

Bruno Hernández Zamora

Integrating Base Stations with a Software Defined Core Network

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 18.04.2016

Thesis supervisor: Prof. Raimo Kantola
Aalto University

Thesis instructor: D. Sc. (Tech.) Jose Costa-Requena
Aalto University

Aalto University School of Science and Technology **Abstract of Master's Thesis**

Author:	Bruno Hernández Zamora	
Title of the Thesis:	Integrating Base Stations with a Software Defined Core Network	
Date:	18.04.2016	Number of pages: IV + 49
School: School of Electrical Engineering **Department:** Department of Communications and Networking		Professorship: S-38
Supervisor:	Prof. Raimo Kantola	
Instructor:	Dr. Jose Costa-Requena	

An unprecedented increase is expected in the demand for mobile data traffic, which requires significant changes in the access network. On the one hand, the penetration of new technologies and networks such as Internet of Things (IoT) introduces a large amount of additional traffic of different kinds. On the other hand, the cell size will be reduced, which results in a higher number of base stations that need to be deployed, requiring additional investments by operators.

This thesis proposes the integration of base stations with a software-defined core network as a way to make the access network more cost-effective, scalable and flexible. Virtualisation technologies allow the possibility of deploying network nodes using generic, off-the-shelf IT hardware, therefore reducing costs thanks to the application of economies of scale. This thesis implements a virtualised solution for LTE network's access node, eNodeB. It will be shown that virtualisation also presents additional benefits in terms of network orchestration and scalability. The result of the thesis can initially be used as a part of a simulated end to end SDN controlled mobile network, allowing the modification of user plane protocol stack. It can be later integrated with a software defined radio component and used with radioheads to form full SDN style base stations e.g. for a 5G test network.

Keywords: virtualisation, LTE, eNodeB, EPC, NFV, SDN, UE

Contents

Acronyms

AM	Acknowledged Mode
ARQ	Automatic Repeat reQuest
API	Application Programming Interface
CCCH	Common Control CHannel
DCCH	Dedicated Control CHannel
DTCH	Dedicated Traffic CHannel
CES	Customer Edge Switching
CMAS	Commercial Mobile Alert Service
DRB	Data Radio Bearer carrying user plane data
DRX	Discontinuous Reception
eNB	eNodeB
EPC	Evolved Packet Core
EPS	Evolved Packet System
ETWS	Earthquake and Tsunami Warning system
E-UTRA	Evolved Universal Terrestrial Radio Access
FMS	First Missing (PDCP) Service Data Unit
GNU	GNU's Not Unix! (Recursive Acronym)
GTP	GPRS Tunnelling Protocol
HARQ	Hybrid Automatic Repeat reQuest
HSS	Home Subscriber Server
IE	Information Element
IMSI	International Mobile Subscriber Identity
I/O	Input/Output
IoT	Internet of Things
ITU	International Telecommunications Union
LTE	Long-Term Evolution
MAC	Media Access Control
MME	Mobility Management Entity
NAS	Non-Access Stratum
NAT	Network Address Translation
NFV	Networks Functions Virtualisation
NSN	Nokia Siemens Networks
OS	Operating System
PCRF	Policy control and Charging Rules Function
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PDU	Protocol Data Unit
P-GW	Packet data network-GateWay
PRB	Physical Resource Block
PPS	Packets Per Second
RLC	Radio Link Control protocol
RRC	Radio Resource Control protocol
S1AP	S1 Application Protocol

SAE	System Architecture Evolution
S-GW	Serving-GateWay
SDN	Software- Defined Networking
SDR	Software- Defined Radio
SDU	Service Data Unit
SRB	Signalling Radio Bearer carrying control plane data
SSH	Secure SHell
s-TMSI	SAE Temporary Mobile Station Identifier
TEID	Tunnel Endpoint IDentifier
TM	Transparent Mode
UE	User Equipment
UM	Unacknowledged Mode
VLAN	Virtual Local Area Network
VM	Virtual Machine
VRF	Virtual Routing and Forwarding

Chapter 1

Introduction

Since their introduction in the market in late 1940's (US) / early 1950's (Europe) [1], the number of mobile phone subscriptions in Europe has grown up to an estimate of 120,6 subscriptions per 100 inhabitants[1] in 2015. Nowadays there is a wide variety of devices in the market, offering advanced technological features and high data transmission rates [2], [3].
This translates to a corresponding increase in the demand of mobile data traffic. The global mobile data traffic grew 81 percent in 2013, and by 2018 it is expected that the aggregate smartphone traffic will be 11 times more than today [4].
As a consequence of this growth, it is expected that new network structures with smaller cell sizes will be deployed. Estimates are that the increase in traffic cannot be supported by the current macro cell networks for another year or two [5]. At the same time, the increase in demand is expected to be coupled together with a corresponding decrease in the price paid for data to operators. These two factors present a future scenario where the elements of the access network need to become cheaper to install and maintain, easier to manage and update, and also more scalable.
In the new paradigm of software defined networks, virtualised access nodes can satisfy these needs, thanks to the advanced state of development of cloud technologies. The proposal of such paradigm is to separate the network elements into forwarding (hardware) and control (software) components. In this way most of the hardware needed to fulfil the network requirements no longer has to be be specialized network hardware. Moreover, the network application servers can be virtualised and sourced on-demand. This separation allows the deployment of Software Defined Network (SDN) which is more dynamic [6] and has the possibility of adding new capabilities or correcting faults in any of the elements in a matter of seconds.

1.1 Motivation

The Evolved Universal Terrestrial Radio Access (E-UTRA) network is the access network of the Long Term Evolution (LTE) architecture which is composed of

[1] (May 2015) ITU statistics: http://www.itu.int/en/ITU-D/Statistics/Pages/default.aspx

commercial eNodeB's that communicate with the core network through mobile-specific protocol solutions. This restricts the flexibility in terms of integration with other network functionalities such as caching, monitoring and packet forwarding.
The virtualisation of the network opens up the possibility of modifying the network protocol stack itself, allowing for a more flexible network implementation. More specifically a new type of base station with SDN capability can be introduced, which is capable of interacting with an SDN-enabled packet core network.
By means of virtualisation and SDN technologies, it is possible to implement different network functionalities on commercial off-the-shelf data communications equipment as opposed to specialized communications equipment. The deployment of more standard, mass-produced equipment provides the possibility of taking advantage of economies of scale and therefore reduces capital expenditure.

1.2 Research problem

The purpose of this thesis is to develop a virtualised eNodeB application and study its performance in the context of a software-defined network, in a practical manner. This application must be capable of utilizing the LTE radio protocols and assign and recognize radio resources, so that it can be later integrated with a software defined radio component and radioheads to form a full SDN style base station.
The application is used to emulate an end to end SDN controlled mobile network, which allows the modification of the user plane protocol stack.

1.3 Objective and Scope

For this thesis I have studied the most relevant functionalities of the eNodeB inside the Evolved Packet Core (EPC). I have designed and implemented virtualised eNodeB and UE applications, which are capable of performing the most significant LTE procedures, so that a user plane connection can be established and used. The logic of both applications is capable of performing basic operations regarding management of radio resources, while at the same time they can be run in a local Ethernet network, without the need of radio heads, for testing purposes.
For the rest of virtualised elements of the network, I have employed an existing implementation of the Mobility Management Entity (MME) and Home Subscriber Server (HSS), as well as an available open source implementation of the Serving /Packet Data Network Gateway (S/P GW).
Finally, I have developed a testbed whose purpose is to measure the performance of the eNodeB application in such a network. The testbed includes a modified version of a traffic analyser that allows the visualisation of LTE protocols and signalling in this context. Different testing configurations have been attempted, and in this thesis I describe the configuration that I considered to present the results that are most reliable and specific to eNodeB. During the testing phase, I have also proposed, implemented

and tested modifications to the user plane stack, the testbed and the applications, to present them as practical examples of the advantages that virtualisation technologies can bring to base stations.

1.4 Structure

This thesis is divided in 7 chapters, where Chapter 1 is the introduction. Chapter 2 provides a theoretical background introducing the concepts of SDN, SDR and presents their respective evolutions in time. Chapter 3 discusses the current development ongoing in parallel with this work, which have been used as reference, as well as the most relevant tools that have been used, and the hypotheses under which the work was made. Chapter 4 describes in detail the practicalities for the implementation of the SDN enhanced eNodeB. Chapter 5 details the experimentation methodology. Chapter 6 evaluates the results, and further analyses the scope and validity of said results. Finally, Chapter 7 provides conclusions.

Chapter 2

Background

This chapter describes the technologies involved in the implementation of the objectives of this thesis. First, a definition of SDN, NFV and SDR technologies is provided, followed by a description of their evolution and relevant examples of application of such technologies. Next, the LTE architecture is described, including its network elements MME and eNodeB. Finally, the LTE protocols involved in this work are described together with their most relevant functions.

2.1 Software Defined Networking (SDN)

SDN is a network architecture that separates the forwarding plane and the control plane functions, which are vertically integrated. The control plane can control several devices[2].
This allows the control to be migrated from individual network elements into remote computing devices. Network administrators can access these remote devices to manage network services, thus allowing those services to be abstracted from lower level functionality. On the other hand, the applications and network services can treat the network as a logical entity [7].

2.1.1 Evolution of SDN

The concept of programmable networks has been present in the industry for many years already. Throughout this evolution, there have been different approaches to the concept of SDN [8].
The Open Signalling working group was created in 1995, promoting the development of a programmable network, which is easy to escalate. After that, an IETF working group released the specification of General Switch Management Protocol (GSMP), a

[2] Open Networking Foundation: SDN definition. Available online:
https://www.opennetworking.org/sdn-resources/sdn-definition

protocol that allows a controller to access many network functions such as reservation and deletion of switch resources.
In the early 2000's, the researchers explored ways to address the challenges of network management functions, such as predicting or controlling traffic routes. The tendency was to find ways to separate the control and data planes so that the network management tasks could be easier. In 2006 NETCONF was proposed as a management protocol capable of modifying the configuration of network devices. In that same year, the Ethane project defined a new network architecture that could use a centralized controller to manage policy and security in a network [8] [9].
Currently, the two most popular SDN architectures are OpenFlow and ForCES. Both approaches establish the separation between the hardware that takes care of data forwarding and an abstraction layer that is responsible for communication with the controller through a signalling protocol (OpenFlow or ForCES). ForCES (Forwarding and Control Element Separation) establishes also a similar separation, defining the Forwarding Element and the Control Element, which communicate with each other through the ForCES protocol.

2.2 OpenFlow

OpenFlow is one of the leading technologies in SDN which is currently promoted by the Open Networking Foundation (ONF). The ONF was formed in March 2011 and backed up by companies like Microsoft, Google, Cisco or Facebook [10].
It was proposed in 2008 as "a way for researchers to run experimental protocols in the networks they use every day" [11]. OpenFlow takes advantage of the capability of many modern Ethernet switches to separate different flows, which allows the possibility to separate ordinary "production" traffic from "research" traffic that can be used for experimentation.

The OpenFlow feature can be easily implemented to an existing switch, since no hardware change is needed. Therefore, hardware vendors tend to support the possibility of enabling OpenFlow in their devices.

2.3 Networks Functions Virtualisation (NFV)

The aim of Network Functions Virtualisation (NFV) is to make use of IT virtualisation technology to standardize a wide variety of network components into a smaller subset of elements, capable of performing data processing, packet switching and storage at a large scale [12]. These network components could be located in "Datacentres, Network Nodes and in the end user premises" [12]. Virtualisation of network functions is not interdependent with SDN. Thus, network functions can be virtualised without the need of SDN implementation, and the separation of control

and data forwarding planes can be achieved without the need of virtualising the networks functions.
Nevertheless, a better performance is expected by combining NFV and SDN. SDN allows for easier operation and maintenance, while on the other hand NFV provides an infrastructure upon which SDN programs can be run.

2.4 Software Defined Radio (SDR)

The Software Defined Radio Forum defines SDR as "Radio in which some or all of the physical layer functions are software defined[3]". This implies that functions of devices which have been usually hardware are now done by a software program, in some cases even the device itself is substituted by a program.
The same hardware is being used as a base for many different devices, thus the cost of each device is lower than specialized hardware's cost, making it more affordable.

2.4.1 Evolution of SDR

The concept of SDR is not new, but the current state of development of the technology makes it possible for developers to implement functionalities that previously have been merely concepts.
In 1987, Air Force Rome Labs developed a programmable modem intended as an evolution of the ICNIA architecture, consisting of several single-purpose radios that were used as one piece of equipment. The next examples can be found at 1990 and 1995 with systems SPEAKeasy-I and SPEAKeasy-II respectively, developed by AFRL and DARPA. In this case it was a fully programmable radio, including a programmable cryptography chip. The second SPEAKeasy model had a more compact size and therefore more practical. Later, these systems evolved so that the software could be fully independent from the hardware and therefore used in other models [13].
In 1996 the SDR forum was founded, with the intention of advocating "for the innovative use of spectrum, and advancing radio technologies that support essential or critical communications worldwide". The organization played an important role in the development of industry standards, which could allow the software components to be more portable across different hardware vendors [13].
The evolution of computers has made it possible that nowadays the digital signal processing operations required for radio hardware can be performed by software in a personal computer. In 2001 the GNU Radio project was initiated, which is a software-defined radio system distributed under the terms of the GNU General Public

[3] SDR forum web page: http://www.wirelessinnovation.org

License, allowing virtually anyone to use it.

2.4.2 Cognitive radio

Cognitive radio was defined by IEEE and SDR Forum as a "Radio in which communications systems are aware of their environment and internal state, and can make decisions about their radio operating behaviour based on that information and predefined objectives" [14].
In basic software defined radio, important radio properties can be software defined, such as carrier frequency, signal bandwidth, modulation and network access. In the current stage of development, many other functions can be defined by software, such as wireless networking, artificial intelligence, the regulatory support, the business model and even the protocols themselves.
Cognitive radio allows for transmitters to measure the spectrum that is currently in use and which spectrum is available, and by selecting the available spectrum it is possible to avoid mutual interference. Other very useful feature is the automatic adjustment of the transmitted power.

2.5 Long Term Evolution (LTE)

The Long Term Evolution project was initiated in 2004 and the first LTE release[4] was 3GPP Release 8 in December 2008 . Even though at the beginning of LTE development there were no applications that had bandwidth and latency requirements as demanding as nowadays, the industry demanded a new wireless system that was suitable for high data rate, as the UMTS system was considered as more oriented towards voice service. At the time when this thesis has been written, LTE remains the latest extension in the evolution of 3GPP systems.
The term "Evolution" in LTE is also used to refer to its specific radio access network Evolved – Evolved Universal Terrestrial Radio Access Network (E-UTRAN), the core network architecture i.e. System Architecture Evolution (SAE) and its corresponding packet core i.e. Evolved Packet Core (EPC).
The EPC is composed by the Packet Data Network Gateway (P-GW), the Serving Gateway (S-GW), the Home Subscriber Server (HSS), the Policy Control and Charging Rules Function (PCRF), and the Mobility Management Entity (MME).
In the outer limits of the EPC is the P-GW, which provides connectivity with external data networks. Among its functions the P-GW takes care or allocating IP addresses to the User Equipment (UE) and filtering of IP packets (guided by the PCRF) into data streams.

[4] 3rd Generation Partnership Project website:
http://www.3gpp.org/technologies/keywords-acronyms/98-lte

The S-GW constitutes an anchor for mobility during LTE handovers and also for mobility between LTE and other technologies. The uplink/downlink data passes through the S-GW, irrespective of the access network. [15].
The Home Subscriber Server constitutes a user database that contains the information of the subscriber profiles and performs authentication and authorization of the user.
The central node of the core network is the MME. It performs bearer management functions, and it is responsible for the UE idle mode procedure. The MME also performs authentication procedures to check the validity of the UE and provides security keys to the eNodeB [15].

2.5.1 eNodeB

The Radio Access Network or E-UTRAN consists only of eNodeB's. The advantage of this in comparison to UTRAN is that the number of network elements in the uplink /downlink data stream is minimized, resulting in lower round trip times. The eNodeB's communicate with other eNodeB's using the X2 interface.
The eNodeB provides the UE with data and control plane terminations. It is responsible for managing the radio resources for the UE and establishing and managing radio bearers towards the UE. The eNodeB is responsible for collecting radio measurements from the UE, and handover procedures.

2.5.2 Mobility Management Entity (MME)

The Mobility Management Entity is responsible for the signalling related to mobility and security for the E-UTRAN access. The MME is also responsible for the management of resources such as the EPS bearers.
The MME constitutes the end point of the Non-Access Stratum (NAS), in contrast with the eNodeB which merely encapsulates the NAS messages in Radio Resource Control protocol (RRC) messages or S1-Application Protocol (S1-AP) messages and forwards them. The protocol stack is shown in Figure 1 where we can identify the location of the NAS and S1-AP protocols.

2.5.3 Protocols

Figure 1 shows the corresponding protocols for the LTE stack. In the control plane protocol stack, it is important to note that the NAS layer is not present in the eNodeB.
The user information can be transmitted through the usage of the user plane protocol stack once the UE has been connected to the network (see Figure 3 and Figure 5).
Also in the user plane stack, it must be noted that the UDP/IP layer is not present in the radio link between UE and eNodeB, and yet a user plane application running in

the UE side may still be using the UDP/IP layer and expecting UDP datagrams.

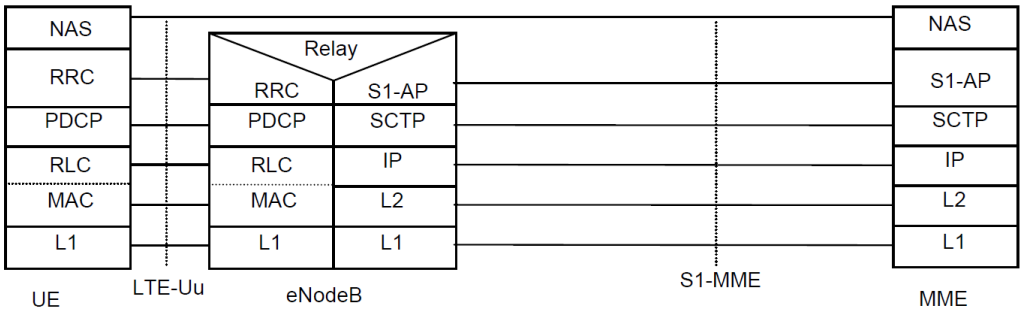

Figure 1: Control plane LTE protocol stacks UE – MME ([16] Fig. 5.1.1.3-1)

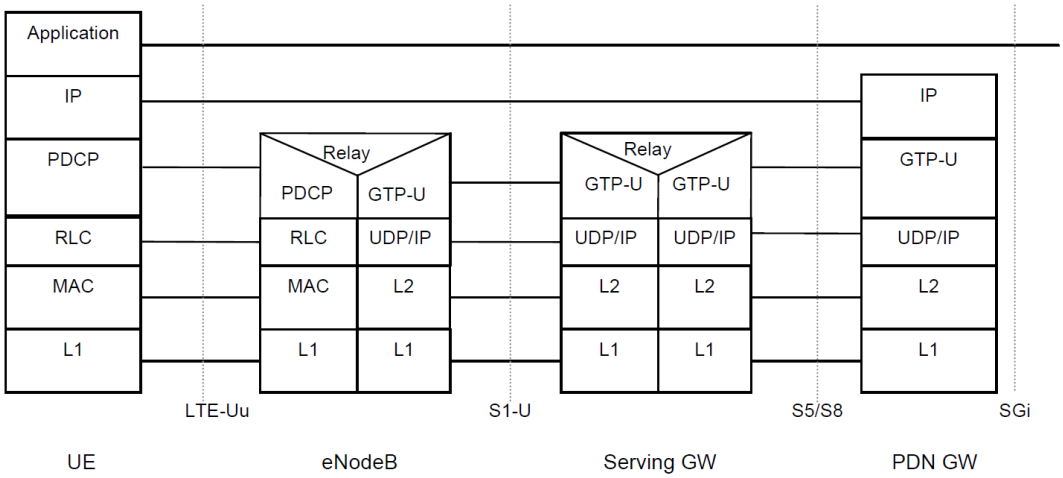

Figure 2: User plane LTE protocol stacks UE – PGW ([16] Fig. 5.1.2.1-1)

2.5.4 X2 Application Protocol (X2AP)

The X2 Application Protocol is the protocol used by eNodeB's to communicate between themselves within the E-UTRAN. It provides the functions of mobility management, load management, reporting of general error situations, resetting of X2, setup of X2, eNB configuration update, mobility parameters management, mobility robustness optimization and energy saving [17].

2.5.5 S1 Application Protocol (S1AP)

The S1 Application Protocol [18] provides the signalling between the E-UTRAN and the Evolved Packet Core, which is required to fulfil a number of functions, including radio access bearer management, mobility functions, handling of information related to active UE and signalling function between UE and MME.
As seen in Figure 1, S1AP will be the highest layer control plane protocol in the eNodeB, meaning that NAS messages directed from UE to MME and vice versa will

be encapsulated within S1AP messages. It also means that the virtualised eNodeB application's logic will not parse NAS messages but simply forward them.

2.5.6 Radio Link Control protocol (RLC)

The RLC protocol is responsible for the framing of RLC Service Data Units (SDU) and putting them into the size indicated by the lower Media Access Control (MAC) layer. These SDUs are then segmented and/or concatenated to construct RLC PDUs [15]. There are three different types of RLC PDUs:
In **Transparent Mode (TM)**, the data is transmitted as it is. No headers are added neither concatenation is performed of the SDUs. TM offers no delivery guarantees [19].
In **Unacknowledged Mode (UM)**, the data is transmitted with RLC headers, but still no delivery guarantees. This is suitable for carrying delay-sensitive user traffic.
In **Acknowledged Mode (AM)**, the data is transmitted with the possibility of retransmission. This mode is used by most of the traffic in both control plane and user plane (background type user traffic such as web browsing and file transfer).
The RLC functions that we considered most relevant with respect to this thesis were TM/UM/AM data transfer and Automatic Repeat Request (ARQ) procedures (polling).
The advantage gained by having the possibility of choosing AM or UM/TM transfer in the RLC layer is that we are allowed to sacrifice reliability in favour of shorter delay by using a non-Acknowledged mode. In the MAC layer, Hybrid Automatic Repeat Request (HARQ) is employed which is faster than ARQ.

2.5.7 Logical channels

Logical channels are established between the sublayers of RLC and Media Access Control (MAC). The differentiation between the different logical channels that exist in LTE has to do with the nature of the information that they transport.
With respect to the control plane, Common Control CHannel (CCCH) and Dedicated Control CHannel (DCCH) are most relevant in this work. The CCCH will be used when an UE has no connection to the access network, while DCCH will be used once the UE is connected.
On the other hand, Dedicated Traffic CHannel (DTCH) will be the channel used for transmission of user plane information, once a data radio bearer is established (see next section).
Other logical channels are: Paging Control CHannel (PCCH) for the paging procedure, Broadcast Control CHannel (BCCH), Multicast Control CHannel (MCCH), Multicast Traffic CHannel (MTCH).

2.5.8 Physical Resource block

The Physical Resource Block (PRB) is the basic radio resource allocation unit for LTE. Different radio signals can be multiplexed depending on their carrier frequency and the time period during which they are sent. Thus, the LTE radio resources consist of carrier frequencies and time slots that can be assigned out of the frequency and time resources available.. The PRB is therefore a unit in bandwidth and time. It consists of 12 subcarriers for a total bandwidth of 180 KHz, and 0,5 ms during which 7 symbols can be transmitted.
Even though this work does not deal with the physical layer directly, the concept of resource block is important for the eNodeB and UE to deduce the corresponding time slot in which the messages are being sent, since this slot will be configured as uplink (UL) or downlink (DL) and thus the message will be considered UL or DL according by that.
Traffic in the uplink direction is sent from the UE to the core network, while downlink traffic is transmitted from the core network to the UE.

2.5.9 Packet Data Convergence Protocol (PDCP)

The PDCP protocol supports functions such as header compression, ciphering and/or integrity protection in the LTE stack. It also has the capability of tracking and maintaining sequence numbers and the order in which the packets are received. In this way, the PDCP layer provides in-sequence delivery of upper layer PDUs.
A PDCP PDU may have 10 different formats if we take into account the different sequence numbers' lengths: these formats are specified in clauses 6.2.2 to 6.2.9 of [20]. These PDUs can be further classified as control or user plane PDUs, being the first type (clause 6.2.2) corresponding to a PDU carrying data from Signalling Radio Bearers (SRB), and the second type (clauses 6.2.3 to 6.2.9) the type that carries data from Data Radio Bearers (DRB). The data plane PDUs themselves can be further classified as Data or Control PDUs, distinction that is indicated in the first bit of each data plane PDU (D/C: 0 for Data and 1 for Control).

2.5.10 Radio Resource Control (RRC)

The RRC layer is present both in UE and eNodeB elements of the LTE architecture. This layer is responsible for many aspects of the radio connection and configuration between both elements and as such it belongs to the control plane.
Because of its presence in the control plane in both of the elements that were virtualised in the development phase of this work, the RRC layer has a wide influence in the architecture and design. It will also play a key role in the design of the API of any software defined module that is involved with the physical layer.
From the point of view of this implementation and upper layers, the important

functions of RRC include the UE measurement reporting, mobility management (handover procedures will be related to UE measurements), transfer of NAS messages and the establishment and maintenance or release of radio connection between UE and eNodeB.

The Radio Bearers (RB) provide a connection segment using the EUTRAN for supporting a LTE bearer.

The Signalling Radio Bearer 0 or SRB0 provides no integrity nor ciphering protection. It is used for the establishment or reestablishment of connection. Only RRC messages are carried through this bearer. Uses the Common Control CHannel (CCCH) on Transparent Mode RLC. The virtualised eNodeB and UE applications will process all packets that are sent through this bearer.

The SRB1 carries most of the RRC messages. It can also transmit NAS information in exceptional cases when SRB2 is not established or when NAS is piggybacked in the Connection Establishment procedure. Uses the Dedicated Control CHannel (DCCH) logical channel on Acknowledged Mode RLC, and PDCP integrity protection and ciphering are applicable.

The virtualised eNodeB and UE application will process packets sent through SRB1 only if the bearer has been configured previously with the radio resources allocated for that UE. The same applies for bearers SRB2, and all Data Radio Bearers.

SRB2 supports lower priority RRC messages. This bearer is typically for uplink/downlink information transfer, carrying NAS information or measurement results. Uses also the DCCH logical channel, so both RLC AM and PDCP ciphering and integrity protection are applicable.

Data Radio Bearers (DRB) 0 to 11 (see [21] clause 6.4) carry user plane messages. UM or AM RLC modes may be used. RRC Connection Setup Complete message may also carry a NAS piggybacked message.

The following RRC procedures are the most relevant ones for this work. Virtualised eNodeB and UE application must be capable of performing these procedures and assign the corresponding radio bearers and logical channels, for the radio emulation to be faithful to LTE specifications.

The **RRC connection establishment** ([21] clause 5.3.3) is used for establishing an RRC connection between EUTRAN the UE (at least SRB1 will be available).

The RRC Connection Setup Complete message can also carry NAS piggybacked messages. This implies that the first NAS message that the UE can send to the eNodeB can be already included in the last message of this procedure.

RRC Messages	RRCConnection Request	RRCConnection Setup	RRCConnection SetupComplete
PDCP	N/A	N/A	
Signalling radio bearer	SRB0	SRB0	SRB1
RLC-SAP	TM - UL	TM - DL	AM
MAC - Logical channel	CCCH	CCCH	DCCH

Table 1: Summary of the RRC connection establishment procedure and the configuration required for each message

The **RRC connection reconfiguration** ([21] clause 5.3.5) can be performed on its own or as part of a larger procedure, for example a handover procedure. The purpose of this procedure is establishing, modifying or releasing Radio Bearers except for SRB1, and/or configure, modify or release the measurements that are to be collected by the UE.
The RRC Connection Reconfiguration Complete message can also carry NAS piggybacked messages.

Event	RRC reconfiguration failure	RRC Connection Reconfiguration Complete
Information element	Without MobilityControlInfo => non- handover	
UE RRC state	RRC_IDLE	RRC_CONNECTED
Other actions	In eNB, after timeout -> initiates RRC connection re-establishment	
PDCP	Release RBs' associated PDCP entity	Available. Depends on configuration
Signalling radio bearer		SRB2
RLC-SAP	Release of RLC entity	Configured
MAC - Logical channel	MAC reset	UL DCCH

Table 2: Summary of the RRC connection reconfiguration procedure and the most important characteristics of each message

The **UL/DL Information transfer** ([21] clause 6.2.2) procedure is used for transmission of NAS messages.
The RRC operations can be modelled by a state machine [22] from the point of view

of the UE (see [21] clause 4.2.1). Each UE may be in the state RRC CONNECTED or RRC IDLE. If supported by the network, the UE may enter also in discontinuous reception (DRX) states.

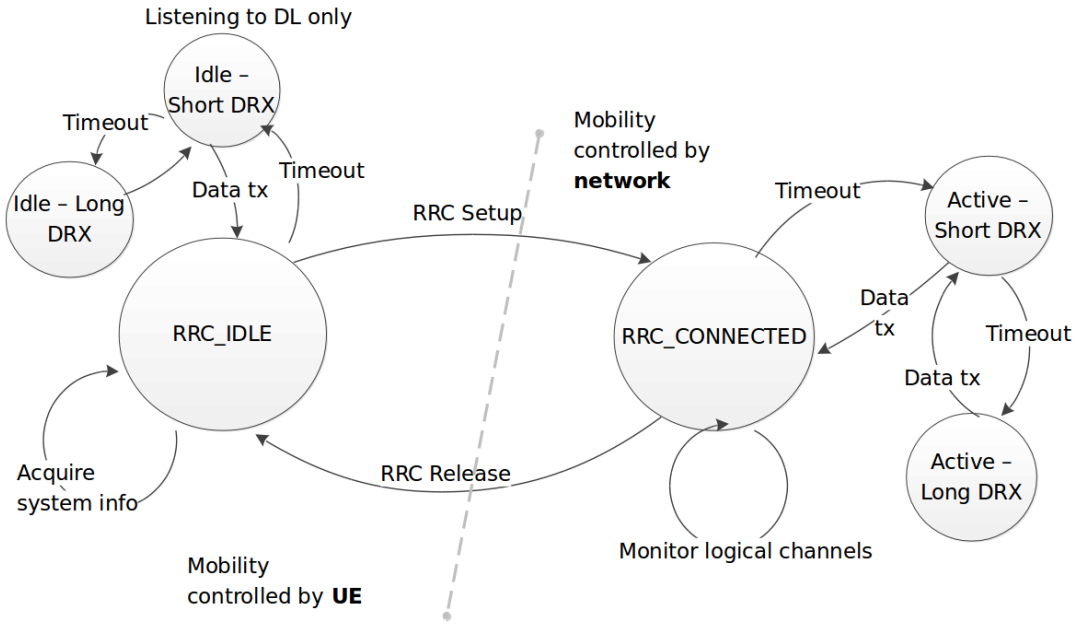

Figure 3: RRC state machine

2.5.11 Control of lower layers in RRC

The procedures presented in section 2.5.10 indicate a degree of control of the RRC layer over the MAC layer. At the same time, procedures corresponding to the MAC layer will require specific parameters from RRC. This can be also observed in the architecture suggestion presented in Figure 4.[23]

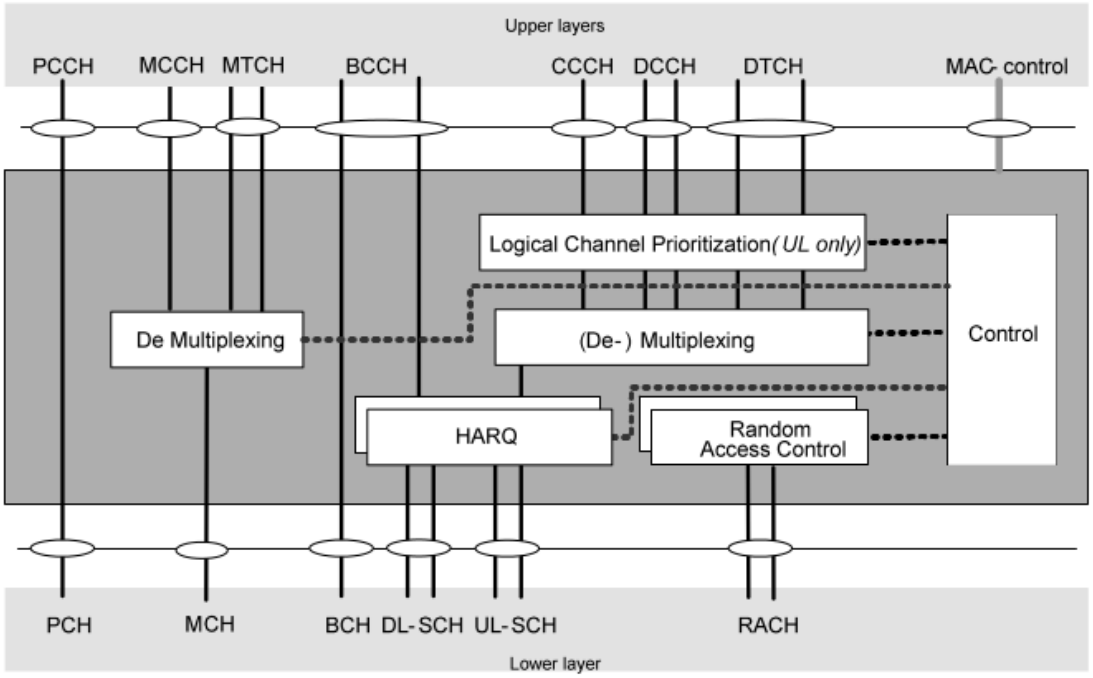

Figure 4: RRC architecture suggestion ([23] fig.4.2.1-1)

According to the definitions given in 2.1 and 2.2, a software-defined network component will be structured in a modular manner with respect to the network layers, and an interface is needed between the different modules. In this case, a software module executing the networking functions of the MAC layer will require certain information elements and parameters from the RRC layer within the time of milliseconds. Thus there must be a fast-access, register-like part of the RRC module where certain RRC parameters are available.
This relationship between the different software modules will later allow the justification of the specific SW eNodeB model developed for this thesis.

2.5.12 Non Access Stratum (NAS)

The Non-Access Stratum is responsible for the attach procedure (Figure 5), which connects the UE to the core network. The NAS is the highest control layer in the LTE stack shown in Figure 1.

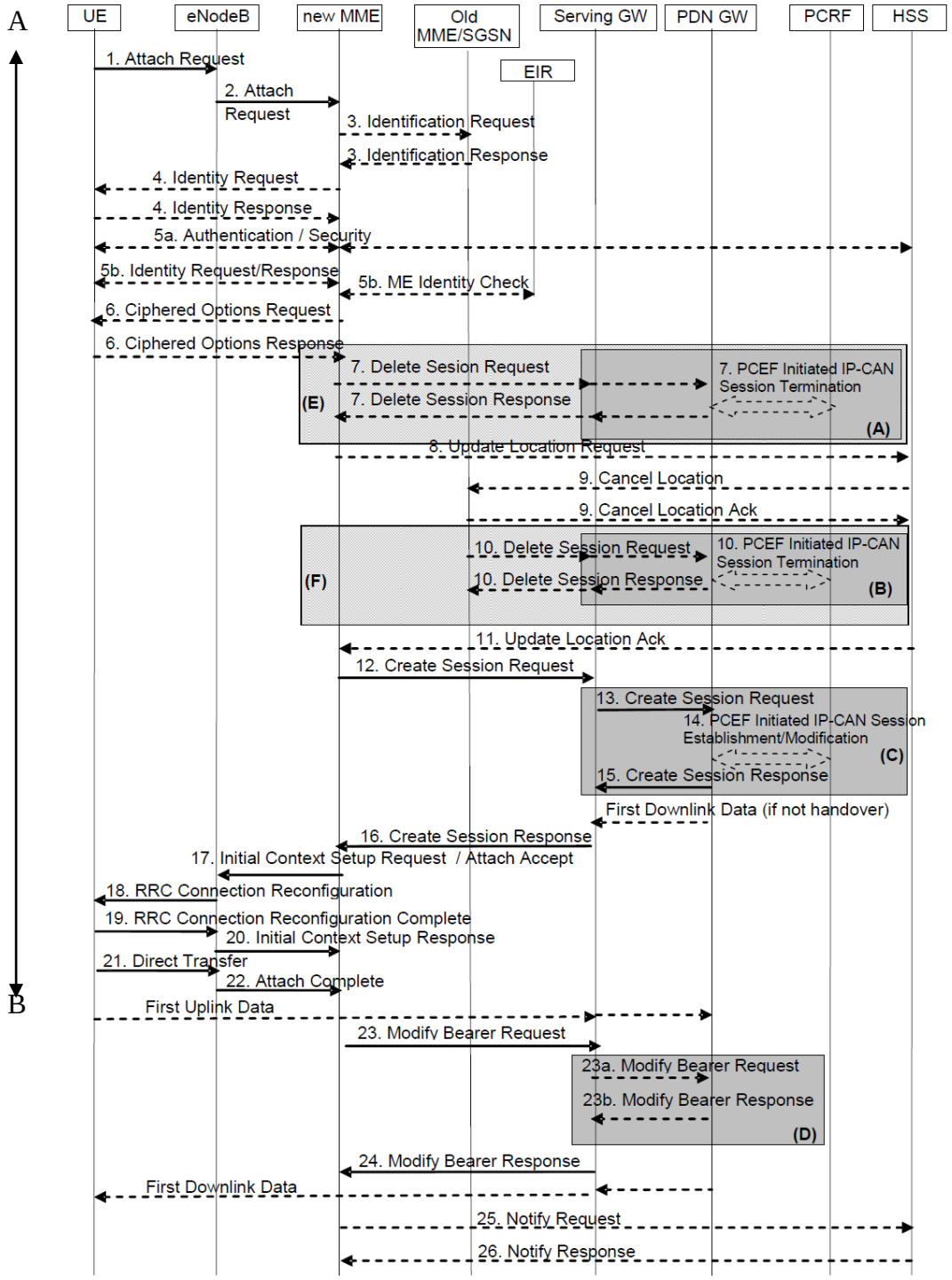

Figure 5: Attach procedure ([16], Figure 5.3.2.1-1)

The attach procedure sets the basic signalling needed for the UE to establish a user plane connection. In this way, this procedure is also later used as a reference for performance in the control plane.

As part of this thesis, a NAS module has been developed for the UE application that connects to the virtualised eNodeB. The implementation of the NAS is functional, and

is capable of connecting and authenticating the UE by generating the parameters in compliance with [24], clause 10.5.3.1 and [25].

Chapter 3

Related work

This chapter presents an overview of the virtualised MME application and its implementation. Next, the GSMTAP header, Virtual Machines and Linux network namespaces are presented, as design tools that will enable the implementation of the virtualised eNodeB and UE application.

3.1 MME model

In October 2013 an experimental EPC was developed in the department of Networking and Communications at Aalto School of Electrical Engineering, using existing UE and eNB emulations provided by a third party company, for which a MME application had to be developed. This MME is now part of the current EPC implementation.
The initial MME design was based on an “engine” part, which is essentially a queue where the states are executed in the sequence defined by a state machine model. This way, it is possible to have different independent modules (state machines) which can be executed in the engine: the S1 module containing the NAS module, S6a module and S11 module.

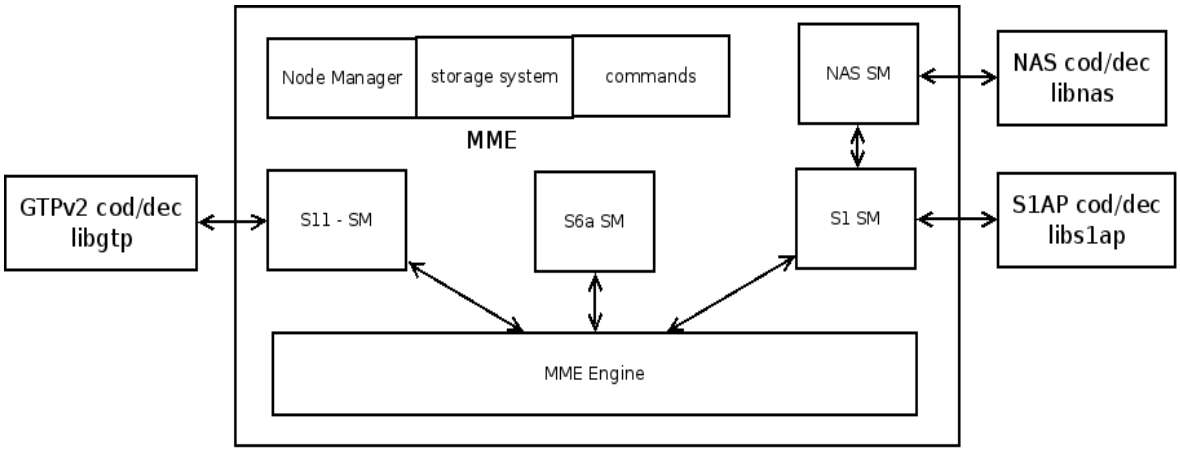

Figure 6: Virtualised MME model [26]

Since its initial proposal in [26], this MME model has been further developed from C based to a C++ based model, providing a more streamlined implementation.
Most of these tasks are done in asynchronous mode, meaning that we are not creating separated threads for each function. This allows for a more scalable solution and also the computing resources are more efficiently used [26].

3.2 Design Tools

A structure as complex as the LTE stack and its functionalities could not be easily virtualised without the usage of a fair amount of existing tools that facilitate the implementation of the corresponding modules in one way or another. In the spirit of keeping compatibility and developing a system as open as possible, tools have been chosen which already were used for similar tasks, or that can facilitate compatibility with industry standards.

3.2.1 The GSMTAP header

GSMTAP is a pseudo-header that is originally intended to transport frames from the GSM radio interface in an UDP/IP message. This header is therefore not part of the protocol itself, and it has been used for free software implementations of mobile interfaces, being OpenBTS (openbts.org) the most notable one. It seems then natural, to continue with this trend when implementing an interface that corresponds to the next evolution of the radio access technology.
On the other hand GSMTAP provides with the specific UDP port 4729 assigned by the Internet Assigned Numbers Authority (IANA). Apart from the advantages derived from having a standardized port to use in cases where no port was otherwise specified, this brings also the possibility of using UDP sockets which will be further discussed in Chapter 5.
The addition of this header allows us to have an abstraction from the physical channels and also of the logical channels if necessary. The GSMTAP header includes a field for the channel type where the channels CCCH, DCCH or DTCH can be indicated (see [27]):

```
/* sub-types for TYPE_LTE_RRC */
#define GSMTAP_LTE_CH_BCCH  0x01

#define GSMTAP_LTE_CH_CCCH     0x02
#define GSMTAP_LTE_CH_DCCH     0x03
#define GSMTAP_LTE_CH_MCCH     0x04
#define GSMTAP_LTE_CH_PCCH     0x05
#define GSMTAP_LTE_CH_DTCH     0x06
#define GSMTAP_LTE_CH_MTCH     0x07
```

Figure 7: Sub-types for LTE RRC defined in the GSMTAP header

In conclusion, the usage of this header is a way to convey parameters that correspond to the radio link, over Ethernet.

3.2.2 Virtual Machine

A virtual machine (VM) is a software program whose purpose is to provide the capability of executing programs as the physical machine would. In the context of this thesis, virtual machines will refer more specifically to virtual computers where the corresponding NFV programs will be run.
The usage of virtual machines provides the possibility of simulating a network scenario with several elements in the same computer. Apart from that, it also allows each element to be running in the most convenient operating system and/or distribution, and the machines can be chosen to be the same that would be used in a real deployment to run the programs, which makes the simulation as realistic as possible, only dependent on the processing power of the host computer.
Virtual machines also makes the simulation independent from the host computer's system and configuration. The NFV software may require specific version of an operating system, kernel, specific libraries or configuration. The employment of virtual machines reduces the configuration requirements of the host computer, since the only requirement is for the computer to be able to run the VM software.
A negative aspect of the VM software is that its performance is expected to be worse that its physical counterpart. Here it is important to note that many of these concepts are not new as explained in Chapter 2, but the current state of technology is what has allowed that the gain in flexibility and reduction in costs can compensate for the loss in efficiency that virtualisation technologies represent.

3.2.3 Linux network namespaces

In the Linux operating system, it is possible to create different network namespaces where each namespace can have its own routing table, its own iptables setup providing NAT and filtering, and run its own network processes. This technology comes from the Virtual Routing and Forwarding (VRF) capability of routers and switches, which has existed since the 1990's and allows, for example, a switch, to support thousands of VLANs at the same time.
In the context of this thesis, the Linux namespaces allow for a more minimalistic approach to the simulation of a network with different elements in the same computer, provided that the corresponding VNF programs can all be run in the same Linux operating system. This tool is therefore convenient for the process of development and testing, since it speeds up the setting up of the simulation and test scenarios.

Chapter 4

Implementation

This chapter describes the practicalities of the implementation of the virtualised eNodeB and the UE application. It begins by presenting an overview of the requirements of the implementation. Next, the software modules are presented, which will perform the functions that correspond to PDCP, RRC, S1AP and NAS layers of the LTE stack. The implementation of each module is explained, as well as the rationale behind the design decisions.

4.1 eNodeB model

The objective of this implementation will be to integrate the virtualised eNodeB in an existing software-defined core network. This implementation must allow the possibility of obtaining measurements that will show the improvements derived from such an integration.
The requirements for scalability have been taken into account in the design of the virtualised eNodeB and all of its modules. As mentioned in section 3.1, the preferred architecture (as advocated e.g. in [28]) is one where only one thread is used (as in one thread per processor core), and all events are handled asynchronously including timers and reading data from sockets.
The model consists of several software modules, named after the LTE layers involved. Each module will perform the functions of its corresponding LTE layer.

4.1.1 PDCP module

After the examination of the protocol stack, it can be noted that the PDCP module has a considerable specific weight in this eNodeB implementation; not only because of its presence and the functionality it provides in the user plane, but also because of its role in the control plane once the RRC connection has been successfully established (which is described in the state machine as “forwarding” state).
I have developed this PDCP module specifically for this project. It is designed in such a way that both UE and eNodeB contain internal PDCP entities that exchange PDCP

PDUs between them, according to the structure proposed in ([20], clause 4.2.1).
Each Radio Bearer is associated with a PDCP entity, which brings that each entity is associated with either user plane or control plane and a bearer index.
Each PDCP entity is responsible for the formatting and maintenance of sequence numbers.
Each PDCP entity is also responsible for the correct ordering of received SDUs (from RLC) according to sequence numbers. This functionality requires the maintenance of fields where a register is kept of the SDUs that have been received correctly.
This specific set of functionalities, implemented according to [20], is expected to work on top of an Ethernet network, which brings up the question of whether the packets should be sent and/or received through conventional UDP sockets, raw sockets, or a customized type of socket which implements this particular protocol. To answer this question, we must take into account the before mentioned "specific weight" of the PDCP module in our implementation.
In the case of **raw sockets**, the result would be a module which is receiving all the possible types of packets that are transmitted to a single Ethernet interface, decoding a PDU which can be of one out of 10 different formats (if we take into account the sequence number sizes). This not only increases the possibility of processing the wrong packet as a PDCP PDU, but also arguably decreases the performance of our implementation heavily by forcing unnecessary processing in the receiving entity's part.
The case of implementing a specific type of sockets for this project poses the challenge of decoding PDU's that lack a unique formal type of header as such, but merely begin with an octet where a set of reserved bits and a sequence number field. Since there are different lengths for the sequence number field depending on whether the PDU corresponds to user or control plane, it follows that the field lengths would be different for each PDCP entity. Furthermore, the sequence number itself could have virtually any content, forcing the implementation to be further customized, away from the standard specifications found in [20]. Compliance with standards and interoperability with commercial equipment being some of the primary goals of this project, this second implementation option can be discarded right away.
With the use of GSMTAP headers, the **UDP socket** option appears as the most feasible and practical. Its only demanding requirement is that it includes elements that are inextricably related to the RRC module, and therefore must be updated constantly and as immediately as possible.
In the radio link a PDCP entity is created in both sides of the communication. In this case Ethernet is being used. In the GSMtap header, the "subslot" field is being used for the UL/DL subslot configuration (see 4.1.3) whose values range from 0 to 9, ten possible values out of 256 possible. For this reason a convention is established between eNodeB and UE that the remaining most significant 4 bits be used for specifying the type of PDCP PDU. This convention is not standardized but specific of this implementation, and it facilitates the receiver's PDCP module the type of PDCP PDU that is being received.

4.1.2 State machines

In this project we designed a set of software modules that correspond to the different layers present in LTE, and each of the modules is modelled as a state machine, or else an aggregate of state machines. In the case of the UE, the NAS layer ([29]: clause 5.1.3) and the RRC layer ([21] clause 4.2.1) have already well defined state machines in the corresponding standard definitions. In the case of the rest of state machines and modules, the implementation has been more specific for the project, although still maintaining a degree of compliance with the specifications. For better clarity, the diagrams of said state machines will be presented for each corresponding section.
The software of each module is written in such a way that the specific operations of the layers, together with the parsing of the information elements (IEs), is located in specific libraries, in such a way that is not necessary for the developer to know the details of the implementation itself.
On the other hand the specific instructions to be executed in each state of the state machine, as well as the conditions that trigger each set of actions, are included in separate files, one file corresponding to each state, often using one sentence that defines each instruction or condition, thus being more easily readable and understandable.

4.1.3 RRC module

As explained in2.5.11, the RRC layer in LTE is responsible of MAC functions which has deep implications in its software implementation.
Without going into deep detail with all possible RRC layer parameters, let us take for example the System Frame Number field (SFN), which is updated every 10 milliseconds. The SFN changes in eNodeB and when it does, this information with at least this frequency is needed by the MAC layer.
It is then reasonable to assume that the RRC layer related information must be stored in a readily accessible, “register”-like memory structure (the heap, accessible via function callbacks). See Figure 8 for a proposal of MAC-RRC software API and refer to sections 2.5.10 and 2.5.11 for elaboration on the roles of each channel or module:

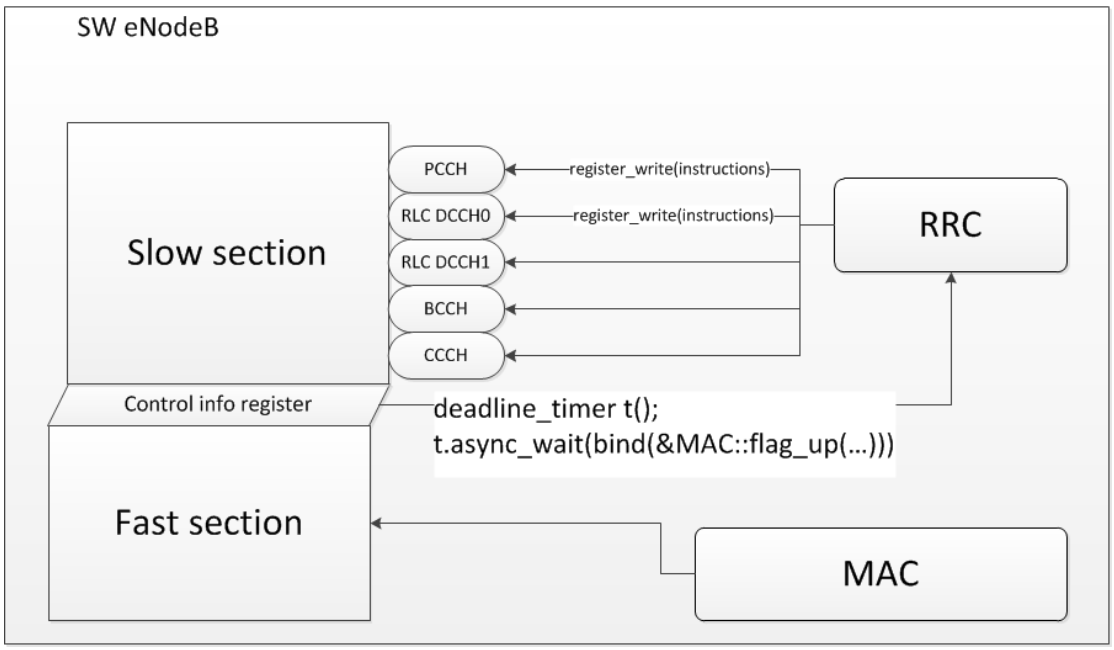

Figure 8: eNodeB MAC API

With respect to data structures related to RRC and in general to all other software modules developed to this thesis such as PDCP, S1AP and NAS, the usage of configuration files has been reduced to the minimum, and for initial configuration purposes (startup).

It is a general practice in cloud applications, that no information is to be stored locally in the computer in runtime, and thus observing this practice in each module keeps the design in line with the cloudification trends (see e.g. [7]: "Future Network Architecture" or [30]: section 4.9).

The different UL/DL configurations that are possible in the radio link between UE and eNodeB are shown in table 4.2-2 of [31]. One of these possible configurations must be used in the radio link, since in each given configuration there is a correspondence between SFN and whether the resource block is UL or DL.

In this implementation, the GSMTAP header is also used to send the SFN value, particularly the 4 least significant bits of the "subslot" field, and configuration 3 from [31] is used by default. This is again a convention used for this implementation only, since a format is needed to convey the type of physical channel that is being used in each case.

4.1.4 RRC state machines

Both the RRC state machine in the UE program and the RRC state machine in eNodeB have the same states that have been presented in section 2.5.10, with the difference being that the eNodeB program will keep track of multiple RRC states machines and sets of RRC timers, one for each UE that is connected.

Due to the nature of the timers that are being used, all the RRC timers are abstracted in an class called "timeline" where all the events are marked. Ultimately only one

"timer" object is used for all UEs, and the "timeline" class is responsible for adjusting the deadline to the nearest event, and updating the list of events afterwards.
The only disparities with respect to the simple state model shown in 2.5.10 are the implementation of the specific RRC connection procedures, which follow the models illustrated in figures 9, 10, and 11..

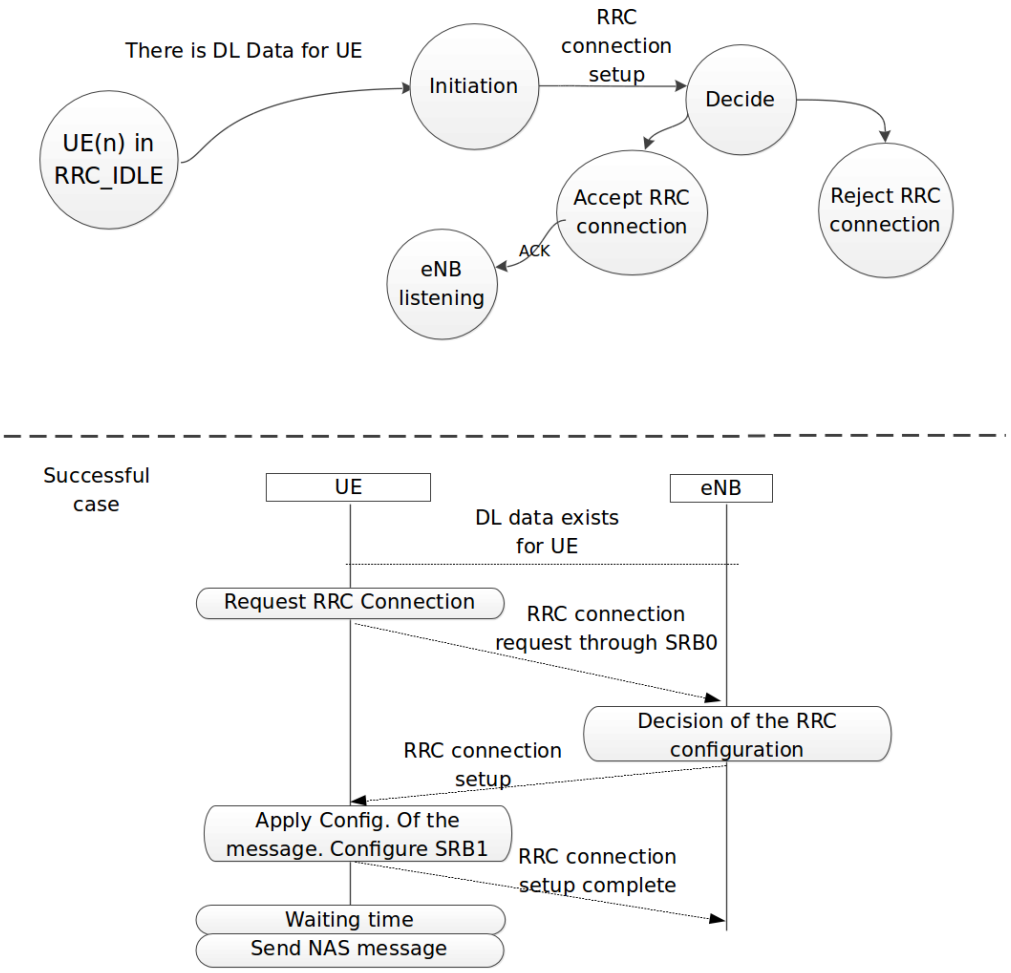

Figure 9: RRC connection procedure

It is important here to remember that the eNodeB does not have a NAS layer module (Figure 1). Therefore, from the eNodeB's point of view, the NAS Attach procedure is simply an RRC Connection Reconfiguration procedure, preceded and followed by a forwarding phase where NAS messages are passed in UL and DL direction encapsulated in RRC (from/to UE) and S1AP (from/to MME) messages.

4.1.5 S1AP module

The model used for the S1AP module and its corresponding state machine is more simple than the RRC model in the sense that only one state machine is considered and

the communication is assumed to be between the eNodeB and one MME (as opposed to multiple UE's in the RRC case).
In the virtualised eNodeB, the S1AP module is the highest level control module. As a consequence of this, the other state machines in the eNodeB will be subject to the S1AP state machine.
It can be recalled from 2.5.5 that the NAS layer is not present in the eNodeB. However, the eNodeB's S1AP module will change its state once the user plane data is enabled, which occurs once the NAS attach procedure is finished.

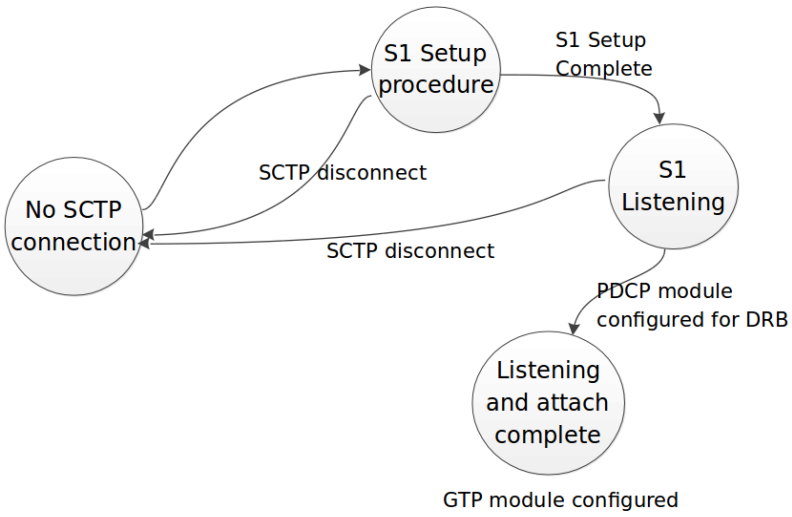

Figure 10: S1AP State Machine

4.1.6 UE NAS module

The UE module has been modelled according to 3GPP standards and is capable of completing a RRC connection procedure, a NAS authentication procedure with the already existing MME and S/P Gateway, as well as UL and DL user plane data transfer once the Attach procedure is finished. The NAS module presented in Figure 11 constitutes a simplified version of the state machine described in (Figure 5.1.3.2.2.7.1 of [29]).

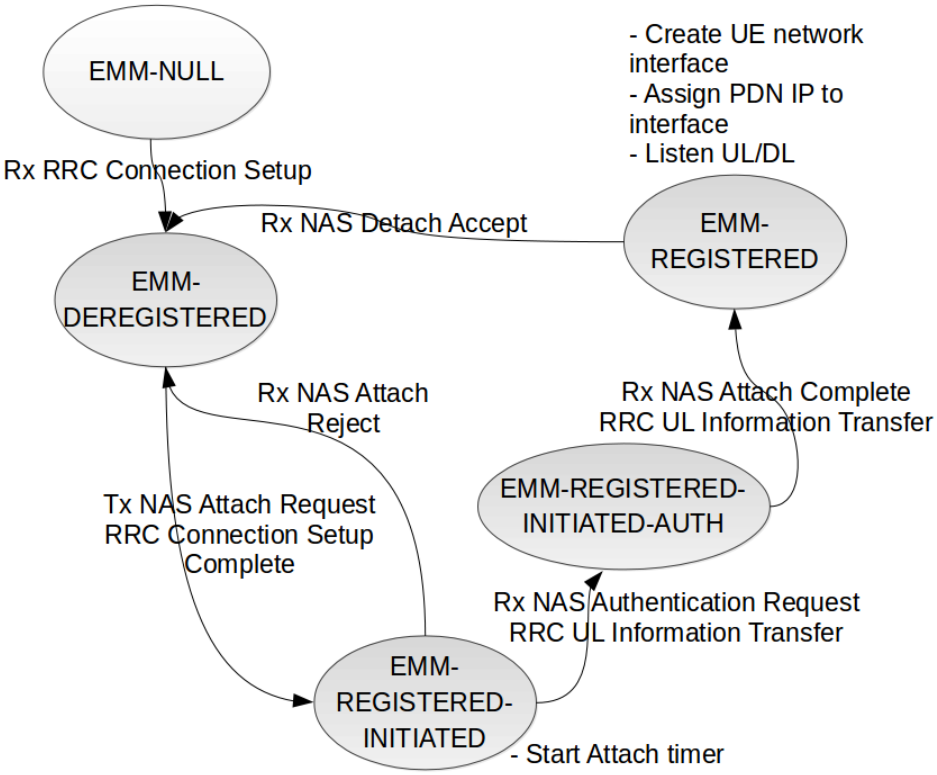

Figure 11: NAS State machine

Chapter 5

Experimentation methodology

This chapter begins by presenting the motivation behind the experimentation. Then it presents the virtual machines and network namespaces scenarios, where different trial simulations have been performed. Each scenario is introduced, followed by a more detailed description of its setup and configuration and a discussion about the challenges encountered when performing a simulation on each scenario. The last section reaches to a conclusion about the scenario from where the test results will be obtained.

5.1 Motivation

For emulating a software defined network with NFV components, some kind of virtualisation technology will be required.
To better understand this requirement, we must note that in the NFV paradigm, each network component's functions are performed by a computer program. This program is designed to be run in a computer with a specific network configuration and which has installed in it the necessary library dependencies for the program to be run. In many cases it is also desirable that the program is running in a specific OS, be it because of advantages of performance that the OS' version provides, or because of specific implementation needs, such a modified version of kernel that includes a customized kernel module (see [32] as an example of this type of implementation).
Therefore is not practical to create a testbed that runs on a specific machine, in a specific operating system. At the same time, the intention behind this work is that other researchers can reproduce it and also that the radio emulation components can be integrated with radio software and a radiohead. It is best to have a software that can be run in different machines, even in cloud computing if desired.
In the sections 5.2 and 5.3, scenarios with different ways of setting up the emulation are proposed and analysed.
Once the system is established, my intention is to test the capacity of the virtualised base station for transmission of user plane traffic. As described in the introduction, it is expected that the base stations will be required to process an ever increasing

amount of user plane data. For this reason, the user plane tests for the eNodeB must involve sending a big amount of user plane packets in the shortest time possible.
At the same time, a simulation involving small packet size will be the most interesting for the purpose of this thesis. 5G technologies are expected to carry a significant amount of machine to machine traffic (M2M) and interactive type of traffic, which involves small packet size (see e.g. section 3.1 in [33]). At the same time, the modification of the protocol stack was mentioned in section 1.2 as part of the objectives of this implementation and it affects small packets the most. The specific type of traffic which was used will be described in detail in the results section 6.2.
Finally, the system under test is the eNodeB. If packets are lost by some element other than the eNodeB, it should not be considered. This means that the network must be configured in such a way that no packets are lost by the kernel.
In the simulations performed to provide the results of Chapter 6, different network measurement tools were used and for any amount of traffic the observed packet loss was 0%. This result makes sense if we consider that the virtualised eNodeB is simply an application with a buffer, which uses the memory of the system. This brings the realization that the system memory and application buffers are big enough not to be a bottleneck for the traffic. For this reason, the eNodeB is considered as a queueing system where packets are not lost, and I will measure its performance in terms of the rate at which it processes packets, also called service rate in queueing theory and denoted by μ. The packet arrival rate or λ is also measured and shown in the results to compare it to μ. The practicalities of how these measurements are taken are described in 6.5.
As an outcome of this testing, the results will show a feasible way of using the application in terms of amount of traffic that it can handle in a period of time. More specifically, we will find an upper bound in the packet arrival rate, which is analysed in detail in section 6.8.

5.2 Virtual Machines scenario

Because of the possibility of many implementation disparities, the most generic experimenting scenario is the one where a number of different virtual machines are run in the same computer or server, each one of the machines running its own networking component in its own OS and/or OS kernel version.
Both the UE and eNodeB applications are being developed in machines with Linux Debian OS. The applications used for MME and S/P Gateway are installed and run in Ubuntu server machines with installations as simplistic as possible, thus the convenience of accessing them through an SSH client instead of the limited built-in terminal. The SSH connection is set up through the “management” network, created with the Virtual Machine management program, which allows to connect to the different VM’s from the host computer’s OS (Windows in this case).
The networking configuration used for this scenario is depicted in Figure 12.

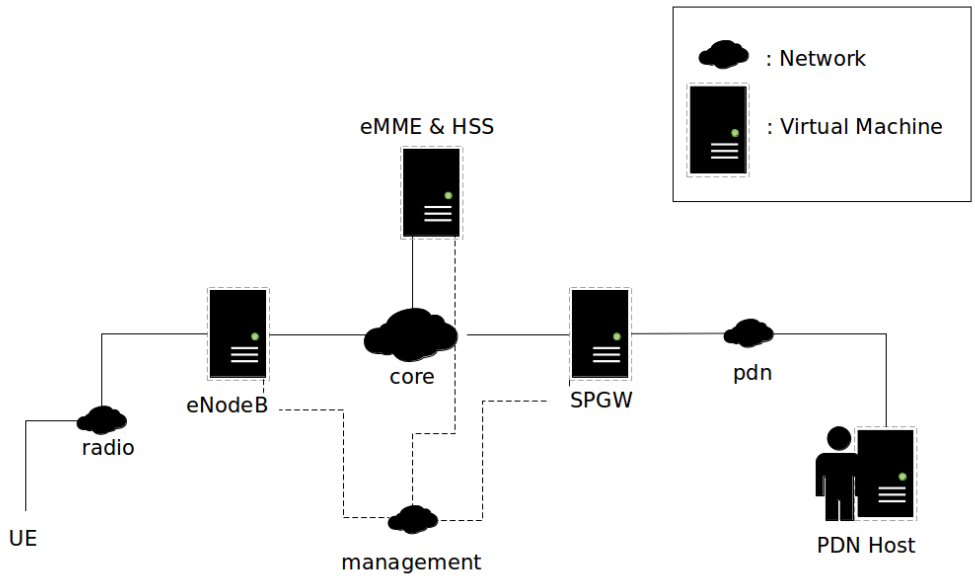

Figure 12: Networks configured in VM scenario

The IP and IP mask of the Packet Data Network (PDN) is set up by the S/P GW and can be configured in the S/P GW machine. The PDN IP that will be assigned to the UE once the Attach procedure is completed successfully will fall within this range of IP addresses.

5.2.1 Challenges of the VM scenario

As mentioned earlier, this scenario allows for the flexibility needed in order to accommodate different implementations which require different operating systems and / or kernel versions. However, this type of experiment with four VM's running in the same computer will impose heavy requirements on the computer's processing power and memory.
Moreover, implementations that rely on a customized version of the kernel will often do so with the objective of improving performance (among others).
A good example on this is the OpenAir interface implementation [32], which includes a latency test in its test suite. At the time when this thesis was developed, OpenAir's requirement for maximum latency in this test was 45000 ns. This requirement was easily fulfilled by one of the lab's machines at Communication and Networking Department in Aalto University, where one of its hard drive's partitions was used to install the specific kernel version required by OpenAir project. However, a virtual machine with the same kernel version and the same installation, run on the same machine, failed to achieve the latency requirement in any of the test runs performed, displaying in some cases more than twice of the maximum latency required. Also the basic installation and execution of the software was noticeably slower in a virtual machine.
In Chapter 6, I will give examples of similar differences in performance my own

implementation.

5.3 Linux networking namespaces scenario

The NFV applications (MME, S/P GW, eNodeB, UE) used in this project have the advantage of flexibility in terms of the variety of Linux distributions in which they can be run. In the VM scenario, different operating systems were used with different networking configurations, but it must be noted that two of them were Debian and the other two were Debian-based distributions.
It is therefore possible to run all four applications in the same Debian machine while keeping the same networks and IP ranges used in the VM scenario, save by the common "mgmt" network which will be no longer needed. This will be accomplished by having 4 different network namespaces, one for each application, and an additional namespace which will represent a client in the PDN side (called "PDN host" in the VM scenario), which connects to the LTE network through the S/P Gateway.
The requirements for this one machine will be having installed the required libraries for all four applications, and also a Linux kernel version of 3.8 or newer, (for full support of SCTP in network namespaces).
The OS will in fact treat these namespaces as if they were different networks, while the improvement in performance will be noticeable especially for slower machines. At the same time, this scenario allows for more flexibility of experimentation, since network namespaces can be deleted and created with one Linux command. The network configuration of this scenario is presented in Figure 13.

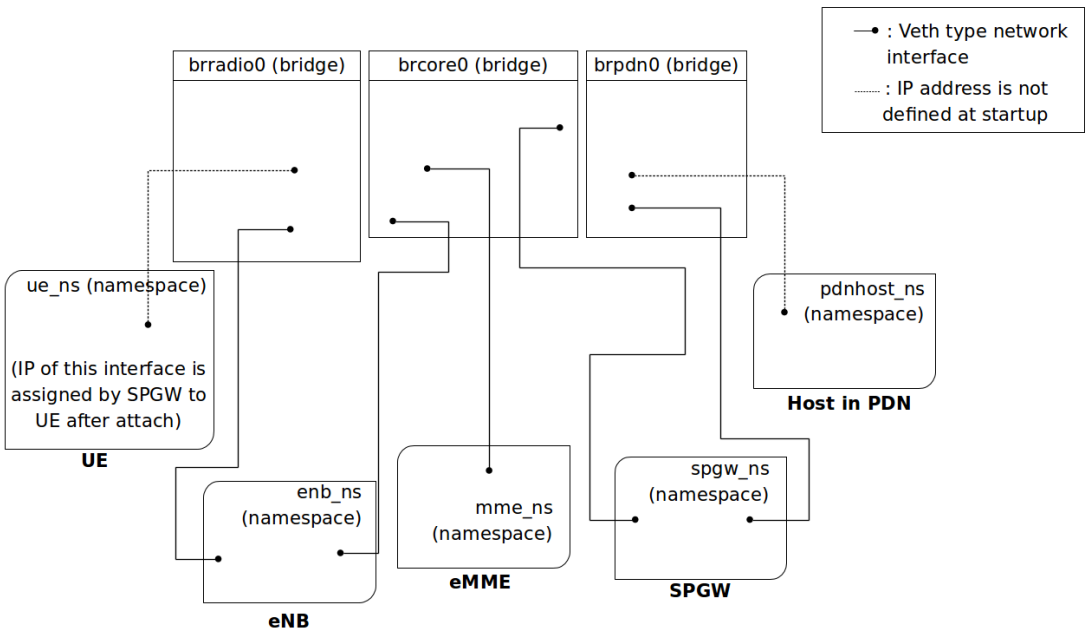

Figure 13: NS scenario's networking configuration

It can be noted that the configuration has remained similar to the one of the VM

scenario, only with the addition of the bridges. Upon the finalization of the Attach procedure, user plane is established and two new interfaces will be created on the UE's namespace. One will be called "uehost", and it will be using the PDN IP that was assigned by the S/P Gateway. The other interface will be called "ueappl", (application interface) and will have no own IP address, but rather a layer 3 socket will be listening bound to this interface, with the purpose of receiving the user plane packets that must be encapsulated and sent uplink to the LTE network.

5.3.1 Challenges of the NS scenario

Apart from the requirement of all elements being run in the same operating system, the results obtained in this simulation scenario presented more stability with respect to the ones obtained with different VM's.
The challenges derived from running this simulation are, for the most part, of practical nature.
Since all of the network elements, tools and services employed must run on the same kernel version with the same base installation, it means that a subset must be found that meets all the requirements.
At the same time, this configuration is vulnerable to new versions of tools and libraries being developed, which render the older obsolete. In extreme cases some components may not be usable anymore in newer architectures (such was the case with e.g. the S1AP module) and tweaks are required. This lack of flexibility was never an issue in the case of the VM scenario, where the same snapshot of the virtual machine is being used independently of the host machine.

5.4 Testing scenario

The trials with different simulation scenarios reveal that it is best to run the emulation in one virtual machine. Using VMs would demand a degree of technical expertise and knowledge of the code that is not reasonable, and only brings the advantage of better performance, which can be also improved by using a machine with more memory and processing power.
On the other hand, the use of multiple VMs harms performance significantly. Once the software can be installed and run, there is no reason not to run all applications on the same VM, with the use of different namespaces. This way, the system can be easily installed in a wide range of systems, and the observations from the emulation results can be reproduced, with differences only in performance.

Chapter 6

Implementation results

This chapter presents the results derived from simulations performed using the virtualised EPC. It begins by defining the scope of the simulation and identifying the factors that will affect the results. Next, the type of traffic that will be transmitted through the EPC is presented. After that, different base systems are described, where the simulations are performed. Then, the different testbed measurements are introduced, as well as the explanation of how the results have been obtained, and the resulting data are presented. After that, the data are analysed in detail, and explanations are provided for the most significant results. The chapter concludes with a comparison of the data obtained from the simulations in different systems.

6.1 Implementation Scope

In order to test the core network and the user plane connection from UE to PDN as a whole, the namespaces scenario has been used. In this way, the results obtained in this simulation are isolated from the effects of the physical network. The packet processing outside the applications themselves is done by the OS's kernel. For this reason, the packet loss (if any) can only be caused by application errors and/or packets lost by kernel. Similarly, the delay is the contribution of the kernel's packet processing delay and the applications'.

6.2 Traffic type

The UDP packets transmitted in the simulations will have a payload size of 12 Bytes. The experiments have been performed with the objective of testing the system's capacity to process big quantities of small packets. The importance of such capacity in the design is explained in section 4.1.1 and motivated by recent technological developments (see [34] section 3.2.1, [35], [30] section 4.9).
Protocols other than UDP may also be used in the testing, but it will result in these packets being encapsulated by UDP packets as soon as the packets are processed by the applications. This can be observed for the eNodeB-MME-SPGW interface in the

user plane protocol stack in 2.5.3. It is also the case for the UE-eNodeB interface when taking into account the usage of GSMTAP header for the PDCP packets described in 3.2.1.

6.3 Base systems

As explained in 6.1, a significant part of the packet processing is performed by the OS kernel. A comparison between packet processing speeds in different kernel versions lies beyond the scope of this thesis. However, the combination of packet processing delay and packet loss imposes a limitation on the virtualisation technology itself which must be accounted for, as will be shown in the results.
The simulations will be run with different base system configurations in order to illustrate the differences in performance between virtualisation technologies.
One simulation will be run in a virtual machine.
A second simulation is run in a virtual machine on the same host machine, whose processor has hardware support for virtualisation enabled. This technology aims to faithfully map the processor instructions in the VM as if they were executed in the native CPU.
A third simulation is run in a Linux container (lxc) over a Linux native OS.
For each result, three different measurements are taken.

6.4 Control plane testbed measurements

The Linux networking stack is complex, and testing it is a study field on itself. This simulation only adds to that complexity by introducing four applications (UE, eNodeB, MME and S/P GW). It is then expected, that on some cases spurious values appear, or that one service fails and data will be missing. The data in table 4 originates from one of many simulations performed, which shows both issues and illustrates the motivation behind this iterative method for obtaining results. In table 4, attach time is calculated as the time it takes from point A to B in Figure 5. CPU usage is provided by CPU timers[5].

[5] More details about accuracy in Boost CPU timers documentation:
http://www.boost.org/doc/libs/1_55_0/libs/timer/doc/cpu_timers.html#Timer-accuracy

VM results	UE 1	UE 2	UE 3	UE 4	UE 5	UE 6
Attach Times (ms):	27,94	33,37	36,09	27,78	32,09	34,11
CPU usage in attach (%)	35,80 %	30,00 %	55,40 %	36,00 %	31,20 %	35,70 %
	UE 7	UE 8	UE 9	UE 10	Average	
Attach Times (ms):	28,03	28,13	30,11	34,19	31,18	
CPU usage in attach (%)	35,60 %	33,20 %	29,20 %	-	35,79 %	

Table 4: Packet loss values from simulation run in VM

The results shown for the average values are consistent relative to the rest of simulations run on the same machine.

As mentioned in 6.1.1, propagation and transmission delays do not apply to this simulation. The results for round-trip time provide a context with other delay values to which they may be compared. Specifically, it is expected to be the longest delay for processing one packet. RTT includes the processing delays of the four applications and the added processing delays of the Linux kernel. It is also run on a VM, which is the least efficient setup for processing time, out of the options presented. The attach timer is implemented in the UE application, and it is started when the UE sends the RRC Connection Setup Complete message to the eNodeB, which has piggybacked the NAS Attach Request. The timer is stopped at reception of NAS Attach Accept by the UE.

6.5 User plane testbed measurements

The objective of this part of the simulation has been testing the user plane connection established between UE and PDN, when exposed to large quantities of small packets in a shortest time possible. It is possible to send packets to the application as fast as possible, but it is not possible to send all of them simultaneously. This situation has been simulated by creating a ping flood, where the PDN IP of the UE is pinged from another host in the PDN.

The flooding with packets is achieved by sending bursts of packets with “zero” delay between them. Here, it is implied that the software itself will not introduce timed delays between each packet that is sent. However, in practice, a minimum delay, caused by the application itself, is always involved between the sending of two packets. For this reason, the network is not expected to become congested immediately as the traffic is sent, but in a progressive manner. The testing process consisted on sending bursts of packets with variable sizes (size of the burst being the number of consecutive, not-delayed packets), and observing the effects in packet loss depending on the amount of packets that is used in each burst.

Considering the complexity of the system that is being analysed, many network measuring tools have been discarded due to their results being misleading. Obtaining

of results may be obstructed by limitations of the measuring tool itself. They can also be affected by specific behaviour of one of the applications, or simply not localized enough to derive conclusions for them. In order to isolate the results obtained in the eNodeB from the rest of the system, four burst timers have been used to count packets and inter-packet times.

The user plane data can be sent in the Downlink direction (from the packet network to the UE), or in uplink direction, (from the UE to the network). The four timers correspond to the four possible combinations of user plane inter-packet times that are Transmitted/Uplink, Transmitted/Downlink, Received/Uplink and Received/Downlink. The resolution of these timers is expected to be close to 1μs [39]. Figure 14 illustrates how the measurements are taken for the downlink direction. The timers register the times of arrival (for example). Then, the difference between consecutive times is stored as inter-arrival time. The eNodeB uses a threshold value for the inter-arrival time to determine what is considered a burst of packets. If the inter-arrival time is smaller than this threshold, the second packet is considered as part of a "burst". When the burst ends or the emulation ends, the number of received packets is divided by the total time duration of the burst to obtain the arrival rate (λ).

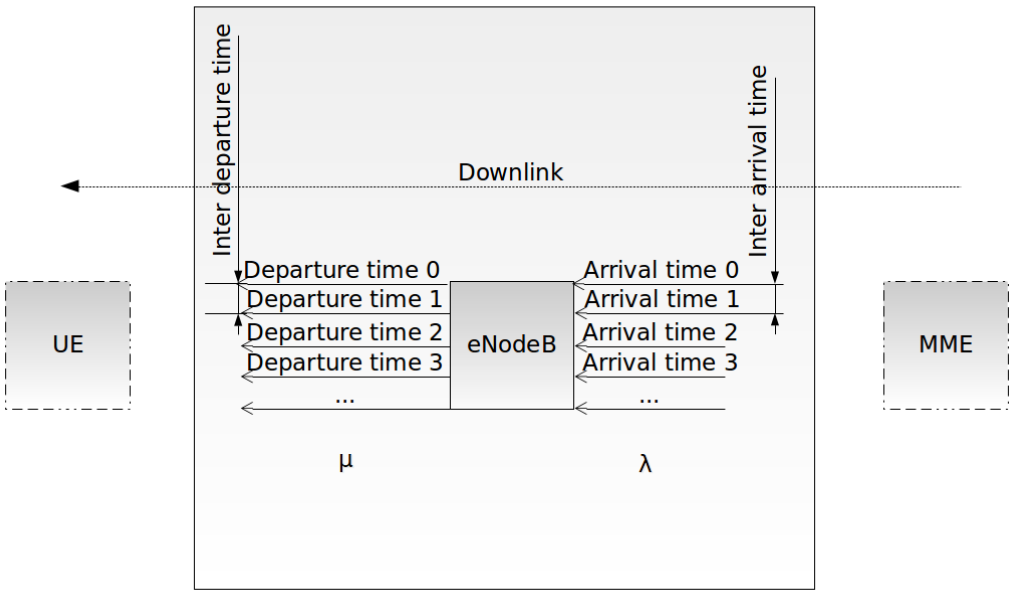

Figure 14: Illustration of how the measurements of arrival rate (λ) and service rate (μ) are taken in the eNodeB application

The purpose of this experimentation is to show a significant difference in the results of one burst size respect to another, which led to the progression of number of burst sizes, shown in the X axis in Figure 15. For this same reason, and due to similarity between results for higher burst sizes (higher than 1000 packets), results are displayed in logarithmic scale.

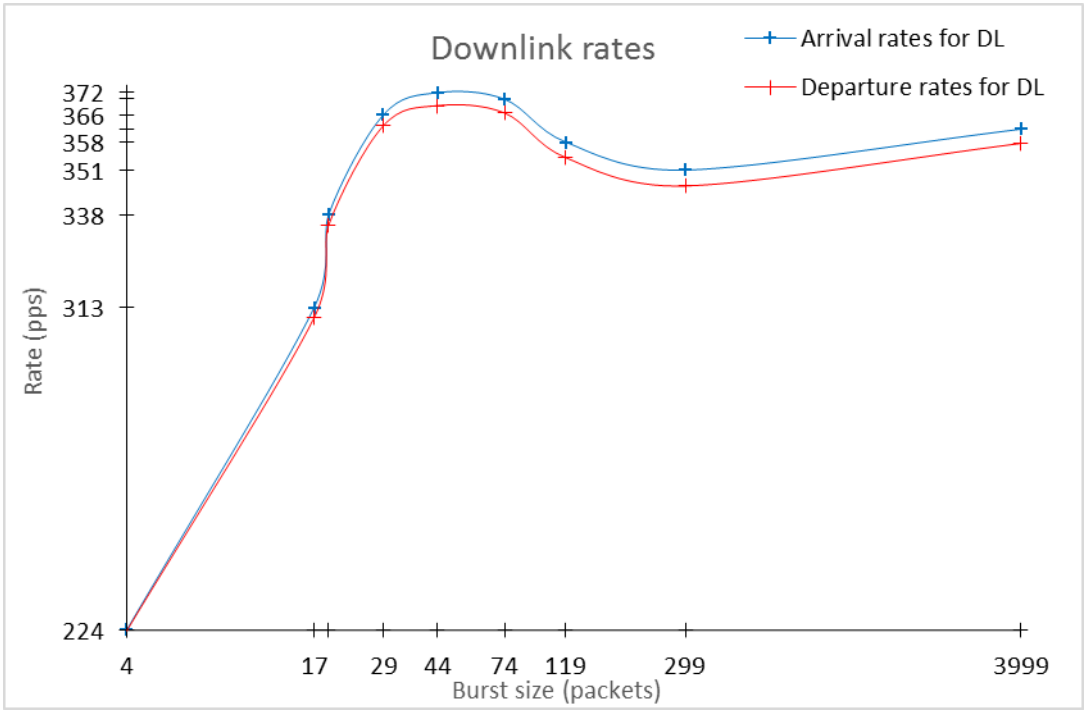

Figure 15: Arrival rate (λ) and service rate (μ) (packets per second) in eNodeB, as a function of burst size (packets), in the downlink direction, from simulation run in VM.

The graph in Figures 14, 15, 16, 17 and 18 is a way to visualise the application's ability to process big numbers of packets, and how many packets can be fit into one burst. However, the x axis in a graph can be easily confused with "time". To avoid this confusion, I have sent burst sizes that increase monotonically with time (waiting 5 seconds between one burst and the next). I have also represented the bursts in chronological order.

Since the application only counts a packet as part of a burst in terms of its inter-arrival time difference with the previous packet, it is easy to deduce from Figure 15 that e.g. the four last bursts that were sent were of 75, 120, 300 and 4000 packets respectively.

In Figure 15 it can be observed that the application is capable of processing packets at similar rates as they arrive, until the arrival rate gets too high (~372 packets per second). For bigger burst sizes, we observe that the arrival rate (λ) decreases, which allows the application to follow up, always at a slightly slower rate than the arrivals.

This decrease in arrival rates limits our ability to test the system, since one of the main objectives of this testbed, as explained in 5.1, is to produce λ high enough to test the limits of the eNodeB application. An assumption is made that such limitation is imposed by the S/P Gw application. This assumption will be explained in more detail in the analysis of section 6.8.

In an attempt to avoid this limitation in arrival rates, a second test will be done where the packets are sent directly to the eNodeB without the intervention of the S/P Gw application.

This test is performed in such a way that, once the attach procedure is completed, the

eNodeB application will assign a port to the connected UE. The testing program that sends user plane traffic to that UE obtains knowledge of that port and the UE's TEID, so the program can add the GTP header itself and send the user plane packets directly to the eNodeB application. In this way, the packets are not processed by the S/P Gw and the inter-packet delay is reduced to a new minimum.

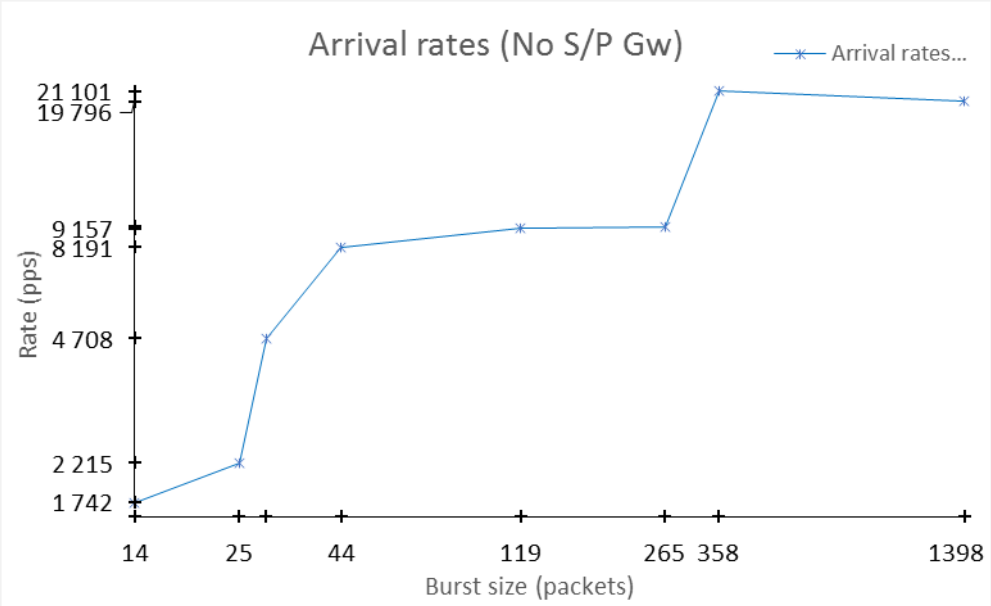

Figure 16: Arrival rate (λ) (pps) in eNodeB as a function of burst size (packets), when packets are sent directly to eNodeB

In Figure 16 we observe that with similar burst sizes we can now create arrival rates now that are much higher, which allows us to stress the application further. However, the curve growth is not completely monotonic with burst size, as we cannot produce higher arrival rates past the 358-packets burst.

It is worth noting that the application has now become unable to process all the packets in the same burst by the time the 299 packet burst arrives. In that case, it records a burst of 265 packets instead of the previous 299 and the remaining bursts get fragmented in the same way. Some of these fragments have been omitted since they didn't produce a high enough λ in the application, which was their initial purpose.

The eNodeB application and the testing program are processes running in the OS, requiring a significant amount of processing time once they reach their maximum rates. The OS manages the computing resources of the machine and distributes processing time among processes with different scheduling priorities. The testing program's priority could affect the arrival rate, and the eNodeB application's priority could affect the service rate. Both testing program and eNodeB application have been run with the same specific scheduling priority[6].

[6] Scheduling priorities set using "nice" program in Linux. See the manual page of "nice" for more information.

With the much higher λ achieved in Figure 16, the transmission rates in the eNodeB will be measured. The rates are expected to be higher, since less processing is now needed for the packets.

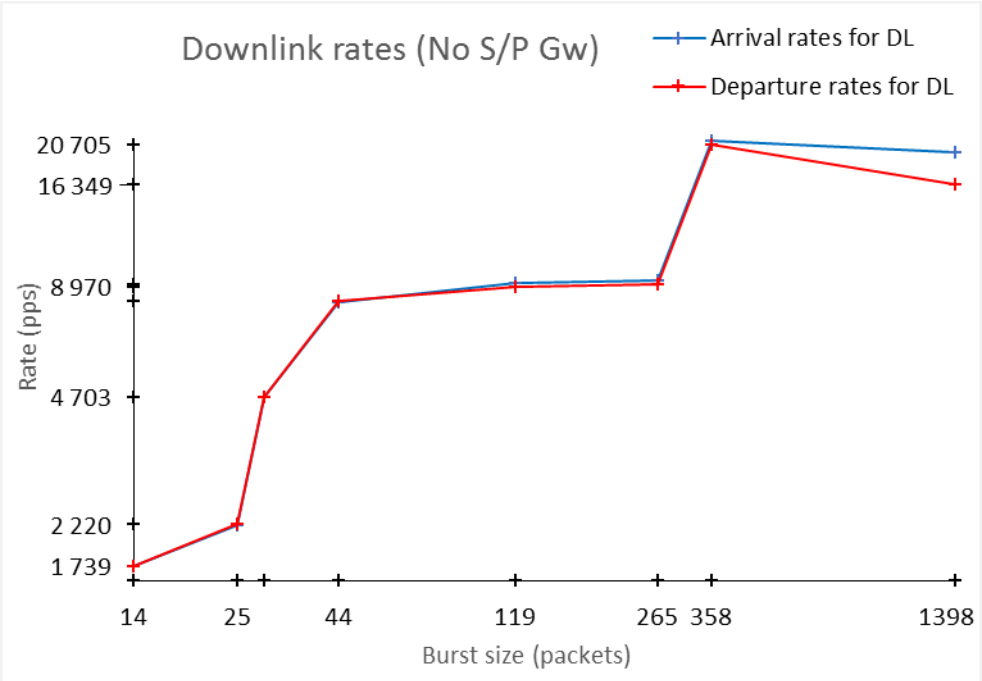

Figure 17: Service rate (μ) (pps) in eNodeB as a function of burst size (packets), from simulation run in VM, when packets are sent directly to eNodeB

In Figure 17, we observe that the application is stable in the range of packet sizes shown in the x axis.

Having the connected UE assigned to a specific port in application runtime, allows for a modification of the eNodeB downlink logic to forward the packet directly to that specific UE. In this way, no additional information is required for the eNodeB logic to identify that UE. Thus the parsing of the UE's TEID value in the GTP header is not needed, and neither is the GTP header itself.

A third experiment is then performed, where modified versions of the packet burst generator and eNodeB applications are used to send the user plane packets without the GTP header.

Figure 18 shows that now the application is stable in a wider range of packet bursts compared to Figure 17.

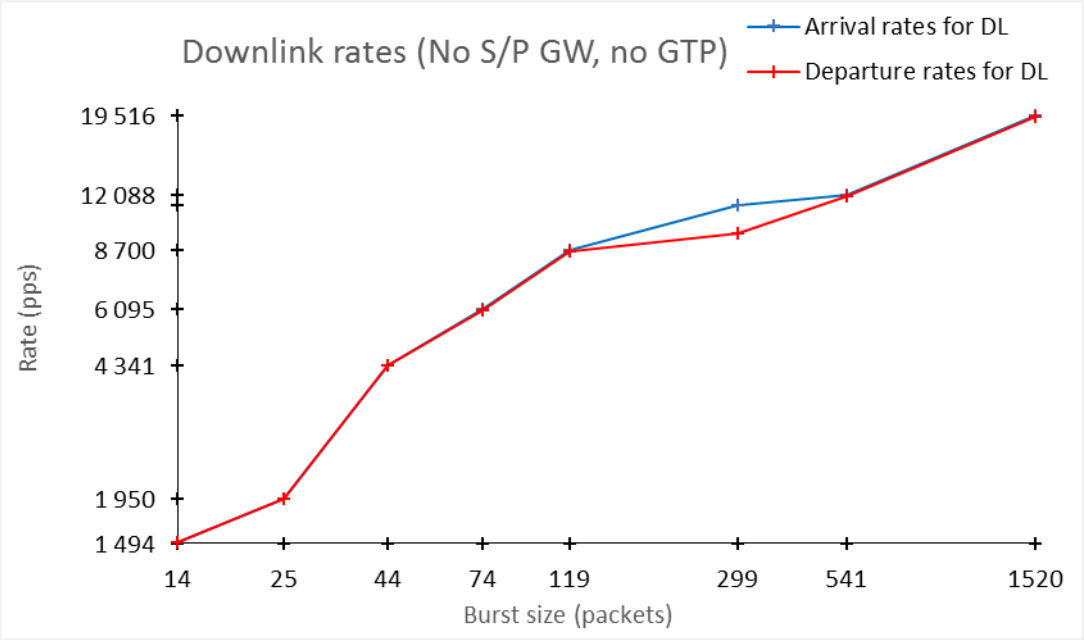

Figure 18: Arrival rate (λ) and service rate (μ) (pps) in eNodeB as a function of burst size (packets), from simulation run in VM, when packets are sent directly to eNodeB and without GTP header.

6.6 Orchestration measurements

Virtualisation technologies provide with the possibility to change the configuration of the system and the network components themselves. The flexibility of the implementation has been tested by organizing two different versions of the components in two branches. These versions differ in the networking configuration used, the interfaces and IP addresses. The eNodeB and UE applications also differ. In the first branch, a less efficient version of eNodeB is used, while UE will have layer 3 sockets both for UL and DL, making the user plane connection much slower for UL. At the same time, the UE implementation makes use of threads. In the second branch, layer 2 sockets are used for UL, UE has a fully asynchronous design which makes no use of threads, more complex configuration is needed and performance is improved.
The process of taking into use the second branch when started from the first branch will be called “upgrade”, and the reverse process will be called “downgrade”. These processes will be timed.
For both directions, the process’ time includes: deletion of old configuration, loading of the new branch, compilation of software in new branch and creation of the new network configuration.

VM	
Downgrade	1 minute 27 seconds
Upgrade	1 minute 26 seconds
VM with virtualisation support	
Downgrade	38 seconds
Upgrade	37 seconds
Linux containers (LXC)	
Downgrade	18,6 seconds
Upgrade	17,1 seconds

Table 5: VM orchestration results

6.7 Results in different base systems

As proposed in 6.3, the same simulations will be run in different systems running on top of the same physical machine. Figure 19 shows results similar in shape to the ones in Figure 15, with all the rates scaled up as a result of the system having more processing power. Figure 19 also shows the same decrease in arrival rates as in figure 15. However, in this case the application has enough processing power so it can keep up with the incoming rate of packets for the whole range.

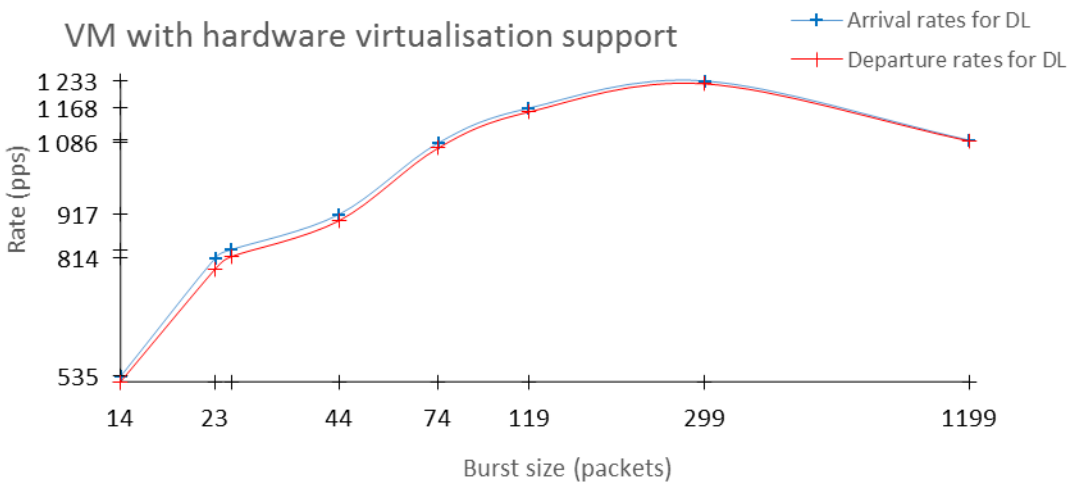

Figure 19: Arrival rate (λ) and service rate (μ) (pps) in eNodeB, as a function of burst size (packets), from simulation run in VM with hardware virtualisation support

6.8 Analysis of user plane tests

The results presented in section 6.5 show the behaviour of a complex system of four software-defined network elements, and will not occur in this network in the absence of said applications.

In Figure 15, the packet bursts present a distribution that hints to the original parameters with which they were sent. For burst sizes bigger than 40, the buffer

limitations of the eNB application begin to show in a gap between arrival and service rates. Two ping floods, of 400 and 4000 packets respectively, were performed at the end of the simulation. Results for higher burst sizes have been omitted for not presenting significant differences. Indeed, as the burst size increases, the arrival rates begin to drop, as they are affected by the slowing down of the preceding applications.
In the asynchronous type of operation, the OS is notified that an input/output operation has to be performed. It is not until the OS notifies that the operation was performed (the packet was read), that the corresponding callback function is run and the corresponding timer value is read. This explains the faithfulness of the values shown for arrival rates. The program continues operation normally and following packets are received when the OS signals it. In this way, packets aren't lost given enough memory, but a higher transmission rate is shown at the point where arrival rate starts to decrease, corresponding to packets from the queue that are still being processed.
The service rates in Figure 16 show the normalizing effect of the application with higher arrival rates. It can be observed that the system is put to its maximum capacity, and packets in the last bursts are being transmitted at a rate which is as high as possible for the application. The application gets close to a maximum service rate and then the rate gets lower as the asynchronous processes wait for the I/O notification from the OS. It is possible that a similar effect takes place in the S/P Gw application, which could explain the difference in smoothness between the curves in Figure 15 and Figure 17.
Figures 16 and 17 present a variability of packet transmission rates. From this we can conclude that there is an upper bound in terms of pps (packets per second), which is the maximum value for λ. Below this upper bound, we have a range of λ within which the application works in a deterministic manner.
In the user plane experiments, the packets are sent with the first 8 octets of the GTP header [36] which are the mandatory part and thus the minimum possible overhead that GTP can add. However, with the payload size of 12 bytes, the total size of the packet is 48 bytes. The GTP header then represents a 16,67% of the total packet size. After removing the GTP header from user plane packets, Figure 18 shows an increase in the burst sizes (X axis in Figure 18) that can be transmitted to the UE compared to Figure 17. In other words, the application is still put to its limit, but the limit is reached less often because more packets can be sent per burst. It also means that the upper bound is now higher, so the range of operation of the application has been increased with respect to the one observed in Figure 18.

6.9 Analysis of orchestration results

The results shown in Table 5 depict the worst possible scenario in terms of knowledge of the target system. The entire configuration must be undone and re-created, and the software must be compiled. There is a possibility that the OS and library versions of the target system are known in advance, in which case the compilation can be done beforehand. This way, 1 minute and 5 seconds can be saved from upgrade times,

which represents 75,6% of the total upgrade time.

6.10 Comparison of base systems

The objective of this base system comparison is showing the effect of hardware features related to virtualisation. Theoretically, it is expected that performance will be better as the system gets closer to the native system.
In Figure 19, the performance improvement allows the application to remain stable for higher burst sizes. That is, there is a small difference between arrival and service rates for the whole simulation, compared to the ~366 pps limit in Figure 15. Moreover, the maximum rates achieved by the application are **3,3 times higher**.
The upgrade time is 37,8 s for VM with hardware virtualisation support, which represents a **43,4% reduction** of the upgrade time.
Finally, the upgrade time for the system installed on Linux containers was 17,1 seconds which is a **45,2%** of the previous time using a VM. This means that, using containers, it took less than half of the time for upgrading the system on the same computer.

Chapter 7

Conclusions

Network Functions Virtualisation is a recent technology, of which few practical examples were known at the realization of this work. It is yet to be seen which ones of the expected benefits of NFV will play a key role in the future networks, or whether it will impact the industry as suggested by [6], [12], [34], and [30].

When a NFV solution is implemented, decisions must be made on which base systems this solution is used. The comparison shown in 6.10 helps illustrate the impact that this decision will have in practice.

The evolution of mobile networks is affected by the appearance of newer systems and technologies. Infrastructures such as the Internet of Things (section 3.2.2 in [37]) are progressively taken into use, populating the network with a much higher amount of small-sized packets. These small-sized packets arrive to the network in such quantity that they cannot be processed in a time interval suitable for the application [38]. Various experiments have been done in this work involving small packets. At the same time, small modifications have been made to improve the performance of the network, serving as an example of the adaptation capabilities of NFV.

Software defined networks offer tangible benefits regarding upgrading, repairing, configuration and overall flexibility of the network. The most significant results in this aspect are the ones shown for upgrade times in 6.9 and 6.10, proving that a new network configuration with enhanced network elements can be set up in a matter of seconds.

Also in section 6.9, two implementations are compared, one of which is fully asynchronous. Here it must be pointed out that this is an extensive study topic, where the validity of the results is dependent on the benchmark that is used. The motivators for an asynchronous event handling approach cannot be solely based on the performance results themselves. However, given the practical nature of this thesis, a case can be made for the ease of implementation and maintenance of the asynchronous model. The complexity in debugging race conditions due to the usage of shared resources increases along with the progress of the development itself. Frequency of appearance of such issues increases with the number of threads and shared resources (either tracked or untracked). The relevance of such factors in modern implementation practices is reflected in the evolution of programming standards themselves, as shown in [39].

Reference

[1] L. Cederquist, M. Ewerbring, G.Sandegren, and J. Uddenfeldt O. Billström. 2006. Fifty years with mobile phones: From novelty to no. 1 consumer product. Ericsson Review.

[2] 2013. Sprint Demonstrates 1 Gigabit Over-the-Air Speed at Silicon Valley Lab. Sprint, press release.

[3] 2014. Nokia Siemens Networks, NSN and Sprint hit 2.6 Gbps TD-LTE throughput. Nokia Siemens Networks, press release.

[4] 2014. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018. Cisco.

[5] White Paper. 2013. Outdoor 3G/LTE Small Cells Deployment Strategy: The Race to the Pole. Nokia Siemens Networks.

[6] White Paper. 2013. Technology Vision for the Gigabit Experience. Nokia Siemens Networks.

[7] WhitePaper. 2012. Software-Defined Networking: The New Norm for Networks. Open Networking Foundation.

[8] A.Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, Thierry Turletti Bruno Astuto. 2014. A survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications surveys & Tutorials.

[9] J. Rexford, E. Zegura N. Feamster. 2013. The Road to SDN: An Intellectual History of Programmable Networks. ACM Queue 11, issue 12.

[10] Steven J. Vaughan-Nichols. 2011. OpenFlow: The Next Generation of the Network. Computer magazine 44, issue 8.

[11] Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner Nick McKeown. 2008. OpenFlow: Enabling Innovation in Campus Networks.

[12] White Paper. 2012. Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action. ETSI.

[13] Bruce Fette. 2009. Cognitive Radio Technology 2nd edition. 2009.

[14] 2007. Cognitive Radio Definitions. SDR Forum.

[15] Sung Duck Chun, Young Dae Lee, Sung Jun Park, Sung Hoon Jung Seung June Yi. 2012. Radio Protocols for LTE Advanced. Wiley.

[16] 3GPP. 2013. General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access. TS 23.401, 3rd Generation Partnership Project (3GPP).

[17] 3GPP. 2014. Evolved Universal Terrestrial Radio Access Network; X2 Application Protocol (X2AP). TS 36.423, 3rd Generation Partnership Project

(3GPP).

[18] 3GPP. 2014. Evolved Universal Terrestrial Radio Access Network; S1 Application Protocol (S1AP). TS 36.413, 3rd Generation Partnership Project (3GPP).

[19] 3GPP. 2013. Evolved Universal Terrestrial Radio Access Network; Radio Link Control (RLC) Protocol specification. TS 36.322, 3rd Generation Partnership Project (3GPP).

[20] 3GPP. 2013. .Evolved Universal Terrestrial Radio Access Network; Packet Data Convergence Protocol (PDCP) specification. TS 36.323, 3rd Generation Partnership Project (3GPP).

[21] 3GPP. 2014. Evolved Universal Terrestrial Radio Access Network; Radio Resource Control (RRC) Protocol specification. TS 36.331, 3rd Generation Partnership Project (3GPP).

[22] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

[23] 3GPP. 2014. Evolved Universal Terrestrial Radio Access Network; Medium Access Control (MAC) protocol specification. TS 36.321, 3rd Generation Partnership Project (3GPP).

[24] 3GPP. 2013. Digital cellular telecommunications system (Phase 2+);Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3. TS 24.008, 3rd Generation Partnership Project (3GPP).

[25] 3GPP. 2014. Universal Mobile Telecommunications System (UMTS); LTE; 3G Security: Specification of the MILENAGE algorithm set; Document 1: General. TS 35.205, 3rd Generation Partnership Project (3GPP).

[26] Vicent Ferrer Guasch. 2013. LTE network Virtualisation. Aalto University, School of Electrical Engineering.

[27] gsmtap.h header file. OSMOCOM Project. http://osmocom.org

[28] Robert Graham. 2013. The C10M Manifesto.

[29] 3GPP. 2014. Non Access Stratum (NAS) protocol for Evolved Packet System. 3rd Generation Partnership Project (3GPP).

[30] 2014. 4G America's Summary of Global 5G Initiatives. 4G Americas.

[31] 3GPP. 2014. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation. TS 36.211, 3rd Generation Partnership Project (3GPP).

[32] OpenAir Project. Eurecom Research Centre, Sophia Antipolis technology park, France.

[33] White Paper. 2014. 5G use cases and requirements. Nokia.

[34] 2014. 4G America's recommendations on 5G requirements and solutions. 4G Americas.

[35] Jesper Dangaard Brouer. 2015. Network stack challenges at increasing speeds. Presentation from Linux Conf Au, New Zealand.

[36] 3GPP. 2013. General Packet Radio System (GPRS) Tunnelling Protocol User Plane. TS 29.281, 3rd Generation Partnership Project (3GPP).

[37] ITU-T. 2012. Overview of the Internet of things. Recommendation Y.2060, International Telecommunication Union.

[38] Chris Snijders, Uwe Matzat, Ulf-Dietrich Reips. 2012. Big Data: Big gaps of knowledge in the field of Internet. International Journal of Internet Science.

[39] Christopher Kohlhoff. 2015. Networking Library Proposal, Revision 4. Targeted for the next ISO C++ standard, C++17.